SYBASE®

XML Modeling

PowerDesigner® 15.3

Windows

DOCUMENT ID: DC20014-01-1530-01
LAST REVISED: November 2010
Copyright © 2010 by Sybase, Inc. All rights reserved.
This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.
To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.
Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.
Sybase trademarks can be viewed at the Sybase trademarks page at *http://www.sybase.com/detail?id=1011207*. Sybase and the marks listed are trademarks of Sybase, Inc. A ® indicates registration in the United States of America.
Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.
Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.
All other company and product names used herein may be trademarks or registered trademarks of the respective companies with which they are associated.
Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.
Sybase, Inc., One Sybase Drive, Dublin, CA 94568

Contents

CHAPTER 1: Getting Started with XML Modeling 1

Creating an XSM 4

XSM Properties 5

Previewing XML Code 8

Customizing your Modeling Environment 9

Setting XSM Model Options 9

Setting XSM Display Preferences 9

Viewing and Editing the XML Language Definition File 10

Changing the XML Language 10

Extending your Modeling Environment 11

Creating Links with Extended Dependencies 12

CHAPTER 2: Building XML Diagrams 13

XML Diagram Objects 14

Linking Objects in an XML Model 16

Creating an XML Diagram 17

External Shortcuts (through References and Data Types) 17

Elements (XSM) 18

Creating an Element 21

Element Properties 21

Linking Child Objects to Elements 26

Manipulating XML Objects Graphically 27

Group Particles (XSM) 29

Creating a Group Particle 30

Creating a Group Particle from the Palette 30

Creating a Group Particle from the Property Sheet of an Element 31

Group Particle Properties 31
Adding a Child Object to a Group Particle 32
Any Elements (XSM) 33
Creating an Any Element 34
Any Element Properties 34
Attributes (XSM) 35
Creating an Attribute 37
Attribute Properties 37
Any Attributes 40
Any Attribute Property Sheet General Tab 41
Constraints: Keys, Uniques, and KeyRefs (XSM) 41
Creating a Constraint 43
Constraint Properties 44
Specifying a Constraint Selector 45
Specifying Constraint Fields 45
XPath Abbreviated Syntax 46
Selector and Field Property Sheet General Tab 48
Groups (XSM) 48
Creating a Group 50
Creating a Reference to a Group 50
Group Properties 50
Linking Child Objects to a Group 51
Attribute Groups (XSM) 52
Creating an Attribute Group 53
Creating a Reference to an Attribute Group 54
Attribute Group Properties 54
Simple Types (XSM) 55
Creating a Simple Type 55
Simple Type Properties 56
Complex Types (XSM) 56
Creating a Complex Type 57
Complex Type Properties 57
Linking a Child Object to a Complex Type 59
Specifying the Type of Content of a Complex Type 60

Derivations: Extensions, Restrictions, Lists and Unions (XSM) 61
Deriving by Extension 61
Deriving by Restriction 62
Deriving by List 66
Deriving by Union 67
Annotations (XSM) 67
Creating an Annotation 68
Annotation Properties 68
Notations (XSM) 69
Creating a Notation 70
Notation Properties 70
Entities (XSM) 70
Creating an Entity 71
Entity Properties 71
Instructions: Import, Include and Redefine (XSM) 72
Creating an Import, Include, or Redefine Instruction 73
Import, Include, and Redefine Properties 73
Redefine Property Sheet Items Tab 74
Business Rules (XSM) 74

CHAPTER 3: Generating and Reverse Engineering XML Schemas and Other Models 75

Generating XML Schema Files 75
Reverse Engineering an XML Schema into an XSM 77
Reverse Engineering to a New XML Model 77
Reverse Engineering to an Existing XML Model 78
Generating Other Models from an XSM 79

CHAPTER 4: Checking an XSM 83

Group Particle Checks 83
Model Checks 84
Data Source Checks 85

Entity Checks **85**
Include Checks **86**
Simple Type Checks **86**
Complex Type Checks **86**
Element Checks **87**
Group Checks **88**
Attribute Checks **89**
Notation Checks **89**
Attribute Group Checks **90**
Import Checks **90**
Redefine Checks **91**
Key Checks **91**
KeyRef Checks **92**
Unique Checks **93**
Extension Checks **93**
Restriction Checks **94**
Simple Type List Checks **94**
Simple Type Union Checks **95**
Annotation Checks **95**

CHAPTER 5: Working with XML and Databases **97**

Mapping Database Objects to an XML Schema Via the XML Builder Wizard **97**
Generating an SQL/XML Query File **99**
Generating an Annotated Schema for Microsoft SQL Server **100**
Generating the SQL Server Annotated Schema File . . 102
Generating an Annotated Schema for Oracle 9i2 ..... **103**
Oracle Extended Attributes for Elements and Attributes 104
Generating the Oracle Annotated Schema File 106
Generating a DAD File for IBM DB2 ..... **107**
DB2 Extended Attributes for Global Elements 108
Generating a DB2 DAD File 108

Index 111

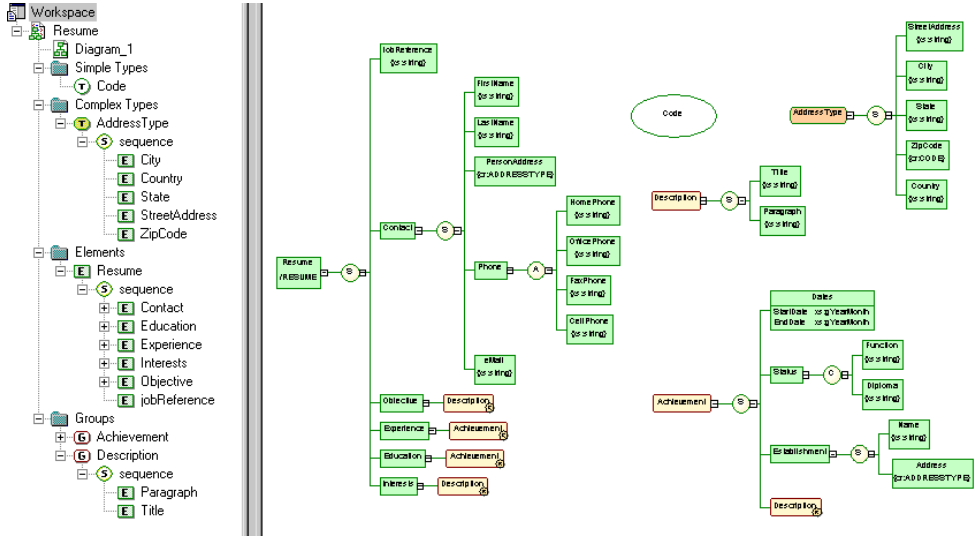
CHAPTER 1 Getting Started with XML Modeling

An *XML model (XSM)* helps you analyze an XML Schema Definition (.XSD), Document Type Definition (.DTD) or XML-Data Reduced (.XDR) file. You can model, reverse-engineer, and generate each of these file formats.

XML (or eXtensible Markup Language) is increasingly used to hold application data because it:

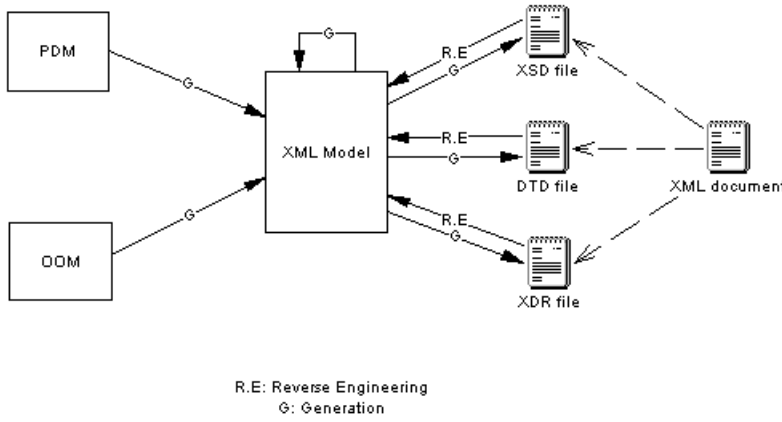
- describes and structures data, whereas HTML only displays data
- uses a self-describing and personalized syntax
- can be exchanged between incompatible systems, since data is stored in plain text format

Since XML structures can be very complex, it is much easier to visualize them through comprehensive and explicit diagrams, than to read XML-coded pages. With its Browser tree view and diagram, a PowerDesigner® XSM gives you a global and schematic view of all the elements composing your XSD, DTD, or XDR:



Once you have created an XML diagram, you can generate an XSD, a DTD or an XDR file from it for use in your application.

A PowerDesigner XSM allows you to generate and reverse engineer XSD, DTD and XDR files and also generate an XML model from a Physical Data Model (PDM), Object Oriented Model (OOM), or another XSM:



DTD, XSD or XDR

The structure of an XSM is described by a DTD, an XSD or an XDR file:

- A DTD file is a basic way to describe the structure of an XML document. It is a raw list of all the legal elements making up an XML document. An extract of a DTD file follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Project Management -->

<!ELEMENT Database (DIVISION,EMPLOYEE,CUSTOMER,PROJECT,
TEAM,MATERIAL,PARTICIPATE,MEMBER,USED,COMPOSE)>
<!ELEMENT DIVISION EMPTY>
<!ATTLIST DIVISION
            DIVNUM                          CDATA
            DIVNAME                         CDATA
            DIVADDR                         CDATA>
<!ELEMENT EMPLOYEE EMPTY>
<!ATTLIST EMPLOYEE
            EMPNUM                          CDATA
            EMP_EMPNUM                      CDATA
            DIVNUM                          CDATA
            EMPFNAM                         CDATA
            EMPLNAM                         CDATA
            EMPFUNC                         CDATA
            EMPSAL                          CDATA>
```

- An XSD file (or schema) is an elaborated way to describe the structure of an XML document. It can support namespaces, derivations, keys, simple and complex user-defined data types and a robust collection of predefined data types. An extract of an XSD file follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
Project Management
-->
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="Database">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="DIVISION">
               <xs:complexType>
                  <xs:attribute name="DIVNUM">
                     <xs:simpleType>
                       <xs:restriction base="ID">
                          <xs:minInclusive value="1"/>
                          <xs:pattern value="00000"/>
                       </xs:restriction>
                     </xs:simpleType>
                  </xs:attribute>
                  <xs:attribute name="DIVNAME" type="NAME">
                  </xs:attribute>
                  <xs:attribute name="DIVADDR" type="SHORT_TEXT">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
```

An XSD file always starts with the <schema> tag (root element). All objects created in the model will appear in the XSD file between the schema start-tag and end-tag

- An XDR file is a simplified XSD file (or schema). It does not support simple and complex user-defined data types. An extract of an XDR file follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="PROJECT"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <description>Project Management</description>
  <ElementType name="DIVISION" content="empty">
     <AttributeType name="DIVNUM"/>
     <attribute type="DIVNUM"/>
     <AttributeType name="DIVNAME" dt:type="string"/>
     <attribute type="DIVNAME"/>
     <AttributeType name="DIVADDR" dt:type="string"/>
     <attribute type="DIVADDR"/>
  </ElementType>
```

An XDR file always starts with the <schema> tag (root element). All objects created in the model will appear in the XDR file between the schema start-tag and end-tag

Suggested Bibliography

- W3C XML Recommendation – *http://www.w3.org/TR/REC-xml*
- W3C DTD Recommendation – *http://www.w3.org/TR/REC-xml#dt-doctype*
- W3C XML Schema Recommendation – *http://www.w3.org/XML/Schema#dev*
- W3C XML-Data Note – *http://www.w3.org/TR/1998/NOTE-XML-data-0105/*

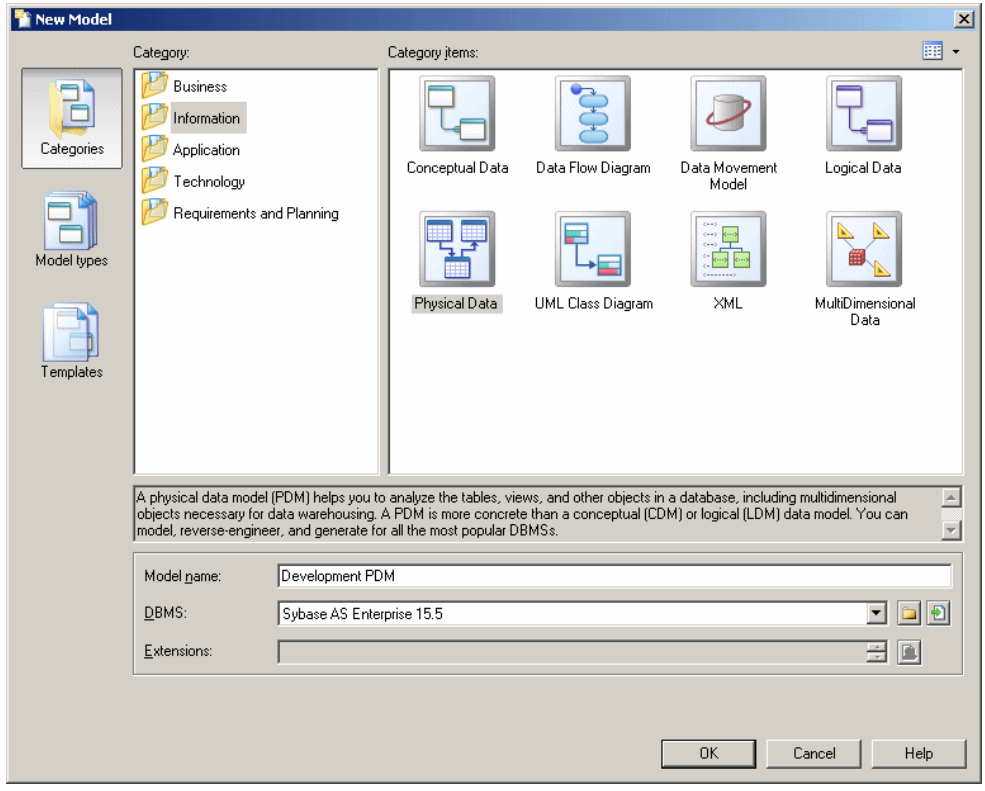
Creating an XSM

You create a new XML model by selecting **File > New Model**.

Note: In addition to creating an XSM from scratch with the following procedure, you can also reverse-engineer a model from an existing XSD, a DTD or an XDR file (see *Reverse Engineering an XML Schema into an XSM* on page 77).

The New Model dialog is highly configurable, and your administrator may hide options that are not relevant for your work or provide templates or predefined models to guide you through model creation. When you open the dialog, one or more of the following buttons will be available on the left hand side:

- **Categories** - which provides a set of predefined models and diagrams sorted in a configurable category structure.
- **Model types** - which provides the classic list of PowerDesigner model types and diagrams.
- **Template files** - which provides a set of model templates sorted by model type.



1. Select **File > New Model** to open the New Model dialog.
2. Click a button, and then select a category or model type (**XML Model**) in the left-hand pane.
3. Select an item in the right-hand pane. Depending on how your New Model dialog is configured, these items may be first diagrams or templates on which to base the creation of your model.

   Use the **Views** tool on the upper right hand side of the dialog to control the display of the items.
4. Enter a model name.

   The code of the model, which is used for script or code generation, is derived from this name using the model naming conventions.
5. Select a target XML language , which customizes PowerDesigner's default modifying environment with target-specific properties, objects, and generation templates.

   By default, PowerDesigner creates a link in the model to the specified file. To copy the contents of the resource and save it in your model file, click the **Embed Resource in Model** button to the right of this field. Embedding a file in this way enables you to make changes specific to your model without affecting any other models that reference the shared resource.
6. [optional] Click the **Select Extensions** button and attach one or more extensions to your model.
7. Click **OK** to create and open the XML model .

Note: Sample XSMs are available in the Example Directory.

XSM Properties

You open the model property sheet by right-clicking the model in the Browser and selecting **Properties**.

Each XML model has the following model properties:

Property	Description
Name	Specifies the name of the model, which should be clear and meaningful, and should convey the item's purpose to non-technical users.
Code	Specifies the technical name of the model, which is used for generating code or scripts.
Comment	Provides a description of the model.
Filename	Specifies the location of the model file. This box is empty if the model has never been saved.

Property	Description
Author	Specifies the author of the model. If you enter nothing, the Author field in diagram title boxes displays the user name from the model property sheet Version Info tab. If you enter a space, the Author field displays nothing.
Version	Specifies the version of the model. You can use this box to display the repository version or a user defined version of the model. This parameter is defined in the display preferences of the Title node.
XML language	Specifies the model target.
Default diagram	Specifies the diagram displayed by default when you open the model.

The following tabs are also available:

- **Detail** - [XSD only] Contains the following properties:

Property	Description
Target Name-space	Specifies a URI as the namespace for all the model objects. All the schema elements with this prefix in their start-tag will be associated with the namespace. For example: http://www.mycompany.com/myproduct/XMLmodel
Language	Specifies the language used in the model. For example: en, en-GB, en-US, de, fr
ID	Specifies the ID of the model. Its value must be of type ID and unique within the file containing the model. For example: XMOD1
Default	Specifies defaults for the **Form** and **Block** and **Final** model object properties.

- **Items** - lists the model's global objects (which have no parent symbol in the diagram, and are directly linked to the <schema> tag). The list reflects the order in which global objects are declared in the schema. You can change the order of declaration by selecting an item in the list and using the arrowed buttons, at the bottom-left corner of the tab, to move it in the list. The following tools are available on this tab:

Tool	Description
	Add Element
	Add Group
	Add Attribute
	Add Attribute Group

Tool	Description
	Add Simple Type [XSD only]
	Add Complex Type [XSD only]
	Add Notation
	Add Annotation [XSD only]

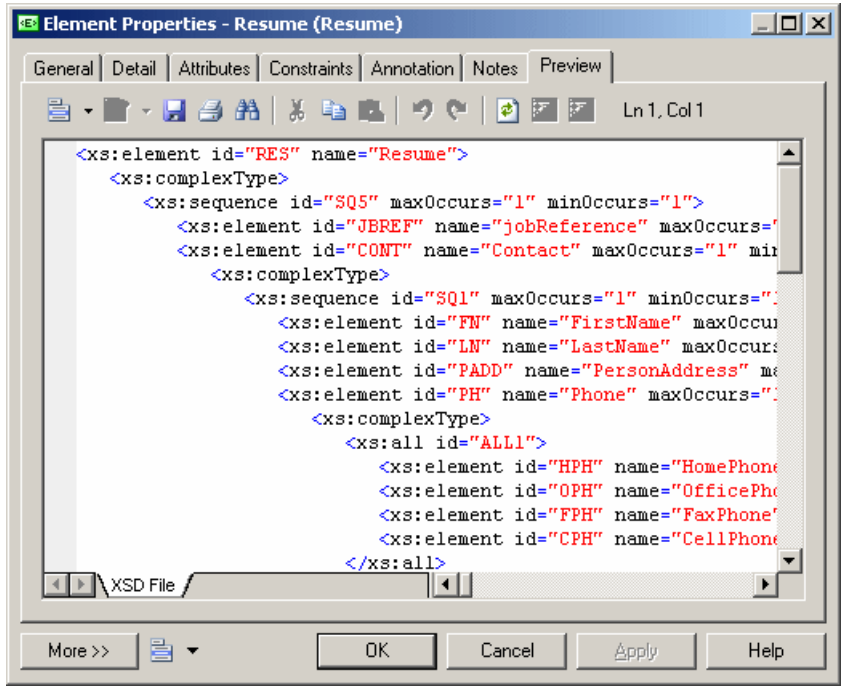
- **External Schemas** - [XSD only] Allows you to link to and reuse in your model global objects from other schemas. The following tools are available on this tab:

Tool	Description
	Add Include
	Add Import
	Add Redefine
	Add Annotation

- **Namespaces** - [XSD and XDR only] Lists the namespaces used to declare objects used in the model.
- **Preview** - Displays a preview of the XSD, DTD or XDR file generated from the XSM.

Previewing XML Code

Click the **Preview** tab in the property sheet of the model, elements, and various other model objects in order to view the code that will be generated for it.



You can use the following tools and keyboard shortcuts from the Preview toolbar:

Tool	Description
	Editor Menu [Shift+F11] Provides access to various editor features, including the insertion of and navigation through bookmarks in the Preview tab: • **Toggle Bookmark [Ctrl+F2]** Inserts or removes a bookmark (a blue box) at the cursor position. • **Next Bookmark [F2]** • **Previous Bookmark [Shift+F2]** Note that bookmarks are not printable and are lost if you refresh the tab.

Tool	Description
	Refresh [F5] - Refreshes the Preview tab. You can debug the GTL templates that generate the code shown in the Preview tab. To do so, open the target or extension resource file, select the **Enable Trace Mode** option, and click **OK** to return to your model. You may need to click the **Refresh** tool to display the templates.
	Select Generation Targets [Ctrl+F6] - Lets you select additional generation targets (defined in extensions), and adds a sub-tab for each selected target. For information about generation targets, see "Specifying Additional Generation Targets" in *Chapter 1, Resource Files and the Public Metamodel* in *Customizing and Extending PowerDesigner* .

Customizing your Modeling Environment

The PowerDesigner XML model provides various means for customizing and controlling your modeling environment.

Setting XSM Model Options

You can set XSM model options by selecting **Tools > Model Options** or right-clicking the diagram background and selecting **Model Options**.

You can set the following options on the Model Settings page:

Option	Description
Name/Code case sensitive	Specifies that the names and codes for all objects are case sensitive, allowing you to have two objects with identical names or codes but different cases in the same model. If you change case sensitivity during the design process, we recommend that you check your model to verify that your model does not contain any duplicate objects.
Enable links to requirements	Displays a Requirements tab in the property sheet of every object in the model, which allows you to attach requirements to objects (see the *Requirements Modeling* guide).

For information about controlling the naming conventions of your models, see "Naming Conventions" in *Chapter 8, Customizing Your Modeling Environment* of the *Core Features Guide* .

Setting XSM Display Preferences

PowerDesigner display preferences allow you to customize the format of object symbols, and the information that is displayed on them. To set XML model display preferences, select **Tools > Display Preferences** or right-click the diagram background and select **Display Preferences** from the contextual menu.

For detailed information about customizing and controlling the attributes and collections displayed on object symbols, see "Display Preferences" in *Chapter 8, Customizing Your Modeling Environment* of the *Core Features Guide* .

Viewing and Editing the XML Language Definition File

Each XSM is linked to a definition file that extends the standard PowerDesigner metamodel to provide objects, properties, data types, and generation parameters and templates specific to the language being modeled. Definition files and other resource files are XML files located in the `Resource Files` directory inside your installation directory, and can be opened and edited in the PowerDesigner Resource Editor.

Warning! We strongly recommend that you make a back up of the resource files delivered with PowerDesigner before editing them.

To open your model's definition file and review its extensions, select **Language > Edit Current Language**.

For detailed information about the format of these files, see *Chapter 1, Resource Files and the Public Metamodel* in *Customizing and Extending PowerDesigner*

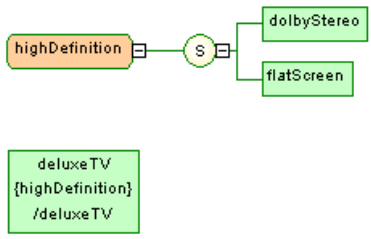
Note: Some resource files are delivered with "Not Certified" in their names. Sybase will perform all possible validation checks, however Sybase does not maintain specific environments to fully certify these resource files. Sybase will support the definition by accepting bug reports and will provide fixes as per standard policy, with the exception that there will be no final environmental validation of the fix. Users are invited to assist Sybase by testing fixes of the definition provided by Sybase and report any continuing inconsistencies.

Changing the XML Language

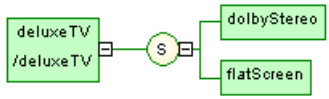
You can change the **XML language** being modeled in your XSM at any time.

Simple types and complex types are only supported by XSDs (schemas). When changing an XSD into a DTD or an XDR, simple types and global complex types (directly linked to the <schema> tag) disappear from the diagram and the Browser tree view. Local complex types (within an element) are expanded in the diagram, beneath their containing element. In this example, HighDefinition is a global complex type, reused as data type for the deluxeTV element:

- In the model with target XSD:

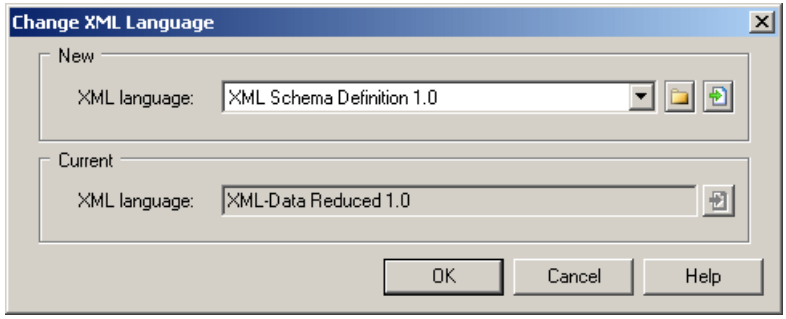


- The model target is changed to DTD or XDR:



Note: You may be required to change the XML language if you open a model and the associated definition file is unavailable.

1. Select **Language > Change Current Language**:



2. Select a **XML language** from the list.

   By default, PowerDesigner creates a link in the model to the specified file. To copy the contents of the resource and save it in your model file, click the **Embed Resource in Model** button to the right of this field. Embedding a file in this way enables you to make changes specific to your model without affecting any other models that reference the shared resource.

3. Click **OK**.

   A message box opens to tell you that the XML language has been changed.

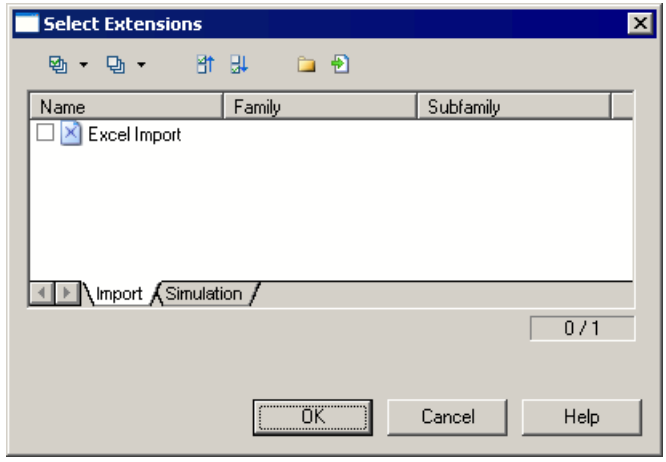
4. Click **OK** to return to the model.

Extending your Modeling Environment

You can customize and extend PowerDesigner metaclasses, parameters, and file generation with extensions, which can be stored as part of your model or in separate extension files (*.xem) for reuse with other models.

To access extension defined in a `*.xem` file, simply attach the file to your model. You can do this when creating a new model by clicking the **Select Extensions** button at the bottom of the New Model dialog, or at any time by selecting **Model > Extensions** to open the List of Extensions and clicking the **Import an Extension** tool.

In each case, you arrive at the Select Extensions dialog, which lists the extensions available, sorted on sub-tabs appropriate to the type of model you are working with:



To get started extending objects, see "Extending Objects" in *Chapter 4, Objects* of the *Core Features Guide* . For detailed information about working with extensions, see *Chapter 1, Resource Files and the Public Metamodel* in *Customizing and Extending PowerDesigner* .

Creating Links with Extended Dependencies

Extended dependencies can be created between any model objects, including between objects in different models. They are used for documentation purposes only, and are not interpreted or checked by PowerDesigner.

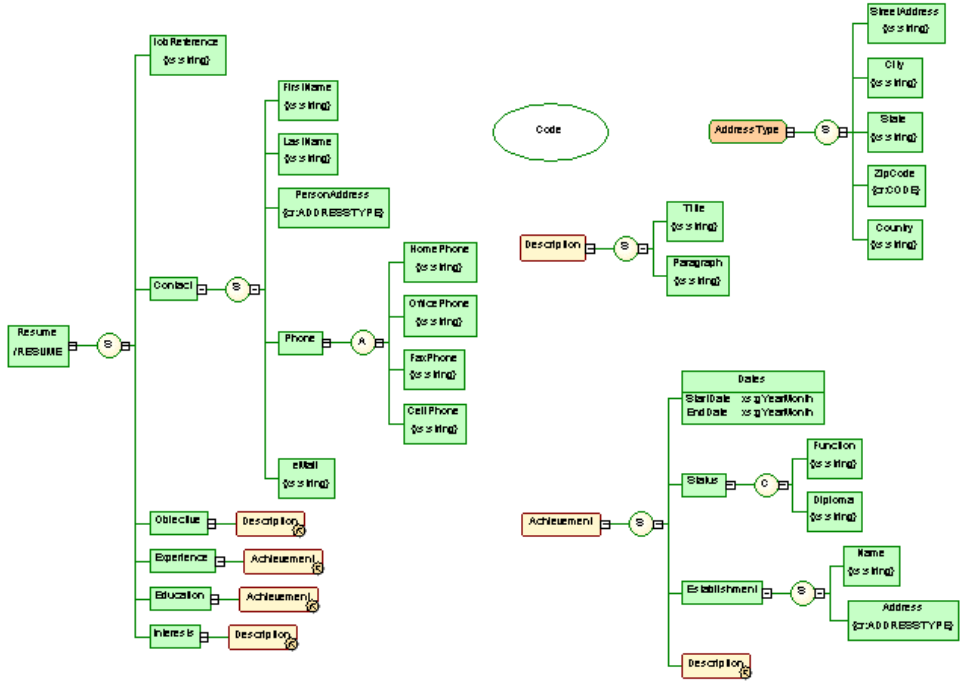
For more information about extended dependencies, see "Creating Extended Dependencies" in *Chapter 9, Getting Started with Linking and Synching* of the *Core Features Guide* .

CHAPTER 2 Building XML Diagrams

An *XML diagram* provides a graphical view of the elements that comprise an XML schema definition in a tree format.

With the user-friendly graphical interface of PowerDesigner XML Model, you can build an XML diagram and then generate automatically an XSD, a DTD or an XDR file.

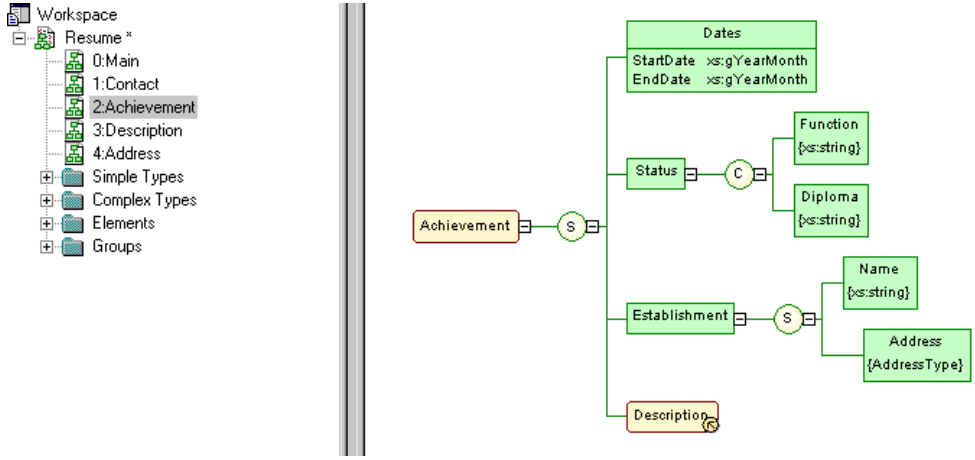
The following example shows the diagram of an XSM which models an XML schema for Resume documents:



If an XML model is too large or too complex, you can create several diagrams to have partial views of the model and focus on certain objects.

For example, the original Resume diagram could be split into five diagrams, corresponding to the five main objects of the model (Main, Contact, Achievement, Description and Address).

The following illustration shows the Achievement sub-diagram:



XML Diagram Objects

PowerDesigner supports all the objects necessary to build XML diagrams.

An XML model represents the structure of a potential or existing XSD, DTD, or XDR through a tree structure of child elements attached to parent elements.

Elements are the basic describing items of an XML model. They can be made of other elements combined in different ways through group particles. Elements are specified by attributes and data types, which can be predefined or user-defined. Simple and complex data types can be defined as global (directly linked to the <schema> tag) or local (embedded in an element declaration).

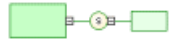
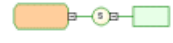
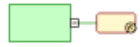
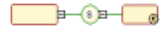
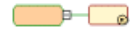
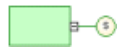
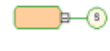
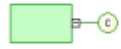
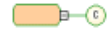
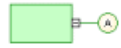
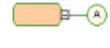
Object	Tool	Symbol	Description
Element		Element /Element	The basic object of an XML model. An element can contain other elements or attributes. See *Elements (XSM)* on page 18.
Any		Any	Any type of object. Can only be attached to a sequence or a choice group particle. See *Any Elements (XSM)* on page 33.
Attribute	N/A	N/A	Additional information about an element or a complex type. Defined by a built-in data type or a simple data type. See *Attributes (XSM)* on page 35.
Group		Group	A group of elements arranged by a group particle. Defined once and reused through references. See *Groups (XSM)* on page 48.

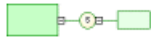
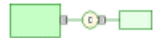
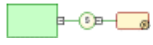
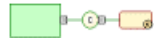
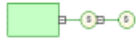
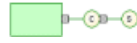
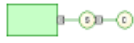
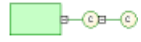
Object	Tool	Symbol	Description
Attribute Group	N/A	N/A	A group of attributes, defined once and reused in the model through references. See *Attribute Groups (XSM)* on page 52.
Simple Type	N/A	N/A	[XSD only] Used in the case of elements or attributes with text-only content. See *Simple Types (XSM)* on page 55.
Complex Type			[XSD only] Used to introduce elements or attributes within an element declaration. See *Complex Types (XSM)* on page 56.
Sequence			This group particle arranges a set of elements, where all the elements must appear at least once in the order of their declaration. See *Group Particles (XSM)* on page 29.
Choice			This group particle arranges a set of elements, from which one element must be chosen. See *Group Particles (XSM)* on page 29.
All			This group particle arranges a set of elements, where each element can appear or not, in any order. See *Group Particles (XSM)* on page 29.
Instruction	N/A	N/A	An import, include, or redefine instruction. See *Instructions: Import, Include and Redefine (XSM)* on page 72
Derivation	N/A	N/A	Extends or restricts the values of elements and simple and complex types. See *Derivations: Extensions, Restrictions, Lists and Unions (XSM)* on page 61
Constraint	N/A	N/A	[XSD only] Specifies uniqueness of element values. See *Constraints: Keys, Uniques, and KeyRefs (XSM)* on page 41
Annotation	N/A	N/A	Provides documentation or application information. See *Annotations (XSM)* on page 67
Entity	N/A	N/A	[DTD only] Specifies a predefined value or external XML or non-XML file. See *Entities (XSM)* on page 70.
Notation	N/A	N/A	Defines and processes non-XML objects within an XML model. See *Notations (XSM)* on page 69

Linking Objects in an XML Model

XML objects do not support standard link objects. To link a child object to a parent object, you must click the child object tool in the palette and then click the symbol of the parent object in the diagram. This will automatically create a link between both objects.

The following tables list the allowed links:

Tool	Element symbol	Group symbol	Complex type symbol
Any			
			
	No link	No link	No link
			
			
All			

Tool	Sequence symbol	Choice symbol	All symbol
Any			No link
			No link
	No link	No link	No link
			No link
			No link
All	No link	No link	No link

Warning! A group particle (sequence, choice, all) cannot be created from scratch in a diagram. It must be the child element of an element, a group or a complex type.

For more information, see *Link Child Objects to Elements* on page 26, *Adding a Child Object to a Group Particle* on page 32, *Linking Child Objects to a Group* on page 51, *Linking a Child Object to a Complex Type* on page 59.

Creating an XML Diagram

You can create an XML diagram in an existing XSM in any of the following ways:

- Right-click the model in the Browser and select **New > XML Model Diagram**.
- Right-click the background of any diagram and select **Diagram > New Diagram**.

To create a new XSM with an XML diagram, select **File > New Model**, choose to create an XML model from the Model type list, choose **XML Model Diagram**, and click **OK**.

Note: The **Symbol > Group Symbols** feature is only available for free symbols in an XML diagram.

Note: Right-click a symbol in an XML diagram and select one of these features: *Expand*: the hierarchy below a symbol is partially expanded (only the first level). *Expand All*: the hierarchy below a symbol is fully expanded (all levels). *Collapse*: the hierarchy below a symbol is hidden. *Arrange Symbols*: the hierarchy below a symbol is properly displayed.

External Shortcuts (through References and Data Types)

External shortcuts allow you to share objects between different models.

You can define external shortcuts in an XML model, but you cannot use them directly in the model, except as substitution groups for elements (see *Element Properties* on page 21).

You can define external shortcuts for any global object (with no parent object in the diagram), except for imports, includes, redefines and annotations .

Internal shortcuts allow you to share objects between packages of a same model. You cannot define internal shortcuts since an XML model does not support packages.

External shortcuts are automatically generated in the following situations:

References

When you use the Reference property to define an element, an attribute, a group or an attribute group, by reference to a similar object in another model opened in the workspace, a shortcut is created between the referencing object and the target object.

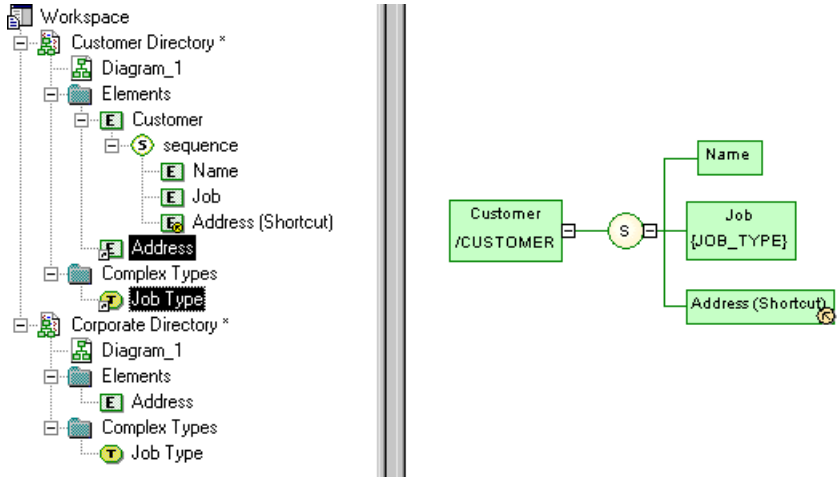
The shortcut is displayed in the current model with a specific item in the Browser tree view and the "(Shortcut)" expression in the reference symbol and item. The target object keeps track of the referencing object in the Reference tab of the Dependencies tab of its property sheet.

Data Types

When you define the data type of an element by selecting a simple or a complex type from another model (using the Browse tool beside the Type list), a shortcut is created between the current element type and the target data type.

The shortcut is displayed in the current model with a specific item in the Browser tree view.

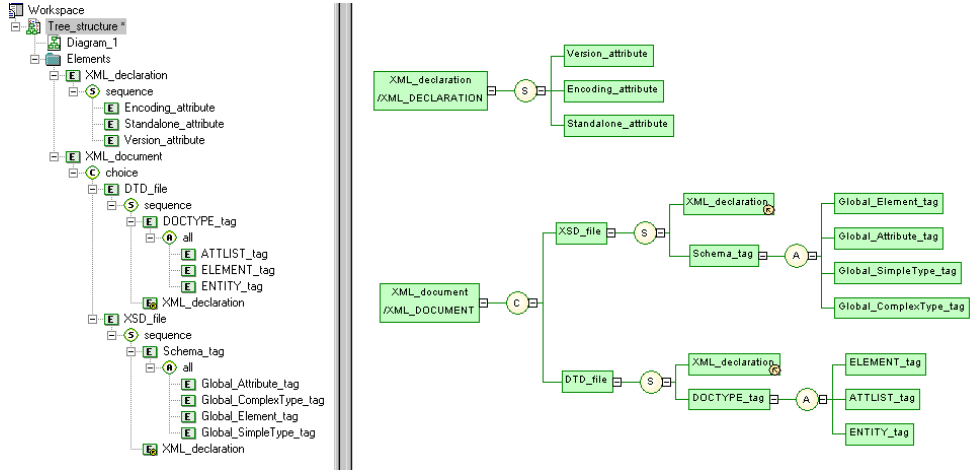
Example of shortcuts through a reference and a data type:



Elements (XSM)

Elements are the basic building blocks of an XML model.

An XML model is a tree structure of elements where child elements are attached to parent elements. For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="XML_DOCUMENT">
      <xs:complexType>
         <xs:choice>
            <xs:element name="XSD_FILE">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="XML_DECLARATION"/>
                     <xs:element name="SCHEMA_TAG">
                        <xs:complexType>
                           <xs:all>
                              <xs:element name="GLOBAL_ELEMENT_TAG"/>
                              <xs:element name="GLOBAL_ATTRIBUTE_TAG"/>
                              <xs:element name="GLOBAL_SIMPLETYPE_TAG"/>
                              <xs:element name="GLOBAL_COMPLEXTYPE_TAG"/>
                           </xs:all>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="DTD_FILE">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="XML_DECLARATION"/>
                     <xs:element name="DOCTYPE_TAG">
                        <xs:complexType>
                           <xs:all>
                              <xs:element name="ELEMENT_TAG"/>
                              <xs:element name="ATTLIST_TAG"/>
                              <xs:element name="ENTITY_TAG"/>
                           </xs:all>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:choice>
      </xs:complexType>
   </xs:element>
   <xs:element name="XML_DECLARATION">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="VERSION_ATTRIBUTE"/>
            <xs:element name="ENCODING_ATTRIBUTE"/>
            <xs:element name="STANDALONE_ATTRIBUTE"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

In a schema, elements are declared with <element> tags.

There are two broad kinds of elements:

- Global elements - have no parent element in a diagram, and are directly linked to the <schema> tag (root element) in a schema. They can be reused in the model through referencing elements (see "XML_declaration" in the example)
- Local elements - have a parent element in a diagram, and are unique within their parent scope. They can be defined by reference to a global element (see the Reference property in *Element Properties* on page 21).

Note: In a model targeted with the XML-Data Reduced language, local elements are first declared separately, like global elements (with the <ElementType> tag and a name attribute), then within their parent element (with the <element> tag and a type attribute).

Extract of an XDR file:

```
<ElementType name="localElement"
<ElementType name="globalElement"
   <element type="localElement"/>
</ElementType>
```

Parent elements are linked to their child elements through group particles (sequence, choice or all), which contain a group of child elements (see the Group type property in *Element Properties* on page 21).

You can derive an XSD element data type to extend or restrict its values (see the Derivation property in *Element Properties* on page 21).

Creating an Element

You can create an element in any of the following ways:

- Use the Element tool in the diagram Palette.
- Select **Model > Elements** to access the List of Elements, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select **New > Element**.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Element Properties

You can modify an object's properties from its property sheet. To open an element property sheet, double-click its diagram symbol or its Browser entry in the Elements folder. The following sections detail the property sheet tabs that contain the properties most commonly entered for elements.

The General tab of an XSD or DTD element property sheet displays the following properties (for XDR element properties, see the subsequent table):

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users.
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces.
Comment	Descriptive label of the element.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.

Property	Description
Reference	Name of a global element. The current element will have the same properties as the global element. The Reference property is only available for child elements. Use the list to select a global element in the current model, or the Browse tool to select a global element from any open model. If you select a global element from another model, a shortcut is created with the referencing element. When you define a reference, name and code properties are grayed. Name and code are those of the global element. Once you have referenced an element, you can locate it in the diagram by right-clicking the referencing element symbol and selecting **Find Referenced Element**.
Group type	Specifies that the element has child elements, and how they are used (see *Group Particles (XSM)* on page 29). You can choose between: • all – All children may be present. • choice – Only one child must be present. • group – Reference to a predefined group (see *Groups (XSM)* on page 48) • sequence – All children must be present in order.
Type	Element data type. Use the list to select a built-in data type. Use the Browse tool to select a simple or a complex type from any model opened in the current workspace. In the case of an XSD, selecting a data type will delete any group particle (and its child elements) or attribute previously defined in the element property sheet. Do not select a data type if you want to define attributes or child elements within the current element
Embedded type	[XSD only] Locally defined data type. It applies to the current element only. Automatically set to Complex if you define a derivation for the element data type.
Content	[XSD only] Content type of the element. If you select *Complex*, the element can have child elements. If you select *Simple*, the element cannot have child elements.
Derivation	[XSD only] Derivation method for the element data type. Used to extend or restrict the values of the element data type. When you define a derivation, the data type disappears. Click the Properties tool to select a base type in the derivation property sheet (see *Derivations: Extensions, Restrictions, Lists and Unions (XSM)* on page 61).

Detail Tab

The Detail tab contains the following properties:

Property	Description
Minimum	Minimum number of times the element can occur. To specify that the element is optional, set this attribute to zero.

Property	Description
Maximum	Maximum number of times the element can occur. For an unlimited number of times, select unbounded.
Substitution group	Name of a global element for which the current element can be substituted. It must have the same type or a derived type. Its value must be a qualified name.
Default	Default value of the element if its content is a simple type or text-only. Enter a default value only if there is no fixed value.
Fixed	Predetermined, unchangeable value of the element if its content is a simple type or text-only. Enter a fixed value only if there is no default value.
Block	Property to prevent another element with the same type of derivation from being used in place of the current element.
Final	Property to prevent derivation of the current element. Prohibited if the element is not a global element.
Form	Form of the element. Used to specify the target namespace of the element. If you select *Qualified*, a namespace prefix is required to qualify the element. If you select *Unqualified*, a namespace prefix is not required to qualify the element.
ID	ID of the element. Its value must be of type ID and unique within the model containing the element.
Abstract	Property defining if the element can appear in the instance document or not. If selected, the element cannot appear in the instance document.
Nillable	Property defining if the element is null or not.

Note: In the case of a model targeted with XDR, the Detail tab is only available for local elements.

Mapping Tab

This tab lets you map the element and its attributes to PDM or OOM objects.

You associate one or more PDM or OOM objects to the element using the **Add Objects** tool on the **Element Sources** sub-tab. You can associate PDM columns or OOM class attributes to the element attributes using following tools on the **Attributes Mapping** tab:

Tool	Description
	Add Mapping - Selects the attributes in the current element that will be mapped to PDM columns or OOM class attributes. Once you have selected the attributes, you can use the list in the Mapped to column to select corresponding PDM columns or OOM class attributes.

Tool	Description
	Create from Sources - Copies PDM columns or OOM class attributes in the data source to the current element attributes.
	Generate Mapping - Automatically generates a mapping between PDM columns or OOM class attributes and element attributes with the same name or code in the data source and the current model.

For more information on complex type mapping, see *Chapter 12, Object Mappings* in the *Core Features Guide* .

The following tabs are also available:

- Attributes - lists the attributes and attribute groups associated with the element (see *Attributes (XSM)* on page 35).
- Constraints - lists the constraints associated with the element (see *Constraints: Keys, Uniques, and KeyRefs (XSM)* on page 41).

XDR-Specific Element Properties

In a model targeted with the XML-Data Reduced language, elements are defined by different attributes:

XDR element attribute	Description
Model	Specifies if an element can contain new local elements. Possible values are: • closed – [default] • open - if an "Any" element is attached to the element. See *Any Elements (XSM)* on page 33 Tab: N/A
Content	Specifies the content type. Possible values are: • mixed - a group particle and a data type are defined • eltOnly - a group particle is defined without a data type • textOnly - a data type is defined without a group particle • empty – neither group particle nor data type are defined Tab: General Field: Group type, Type

XDR element attribute	Description
Order	Specifies how child elements are organized within a parent element. Possible values are: • seq - sequence group particle • one - choice group particle • many - all group particle Tab: General Field: Group type
dt:type	Specifies a data type. Tab: General Field: Type
dt:values	Specifies a list of possible element values. Tab: Values
type	[local elements only] Specifies the name of a global element as reference for the local element Tab: General Field: Reference
minOccurs	[local elements only] To specify the minimum number of occurrences for a local element. Usually set to *0* or *1* Tab: Detail Field: Minimum
maxOccurs	[local elements only] To specify the maximum number of occurrences for a local element. Usually set to *1* or *** (unbounded) Tab: General Field: Maximum

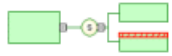
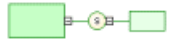
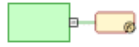
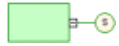
Example of an XDR file:

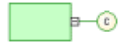
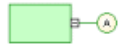
```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_elements"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="localElement" content="empty"/>
  <ElementType name="globalElement" model="closed" content="eltOnly" order="seq" dt:values="0 1 2">
     <element type="localElement" minOccurs="0" maxOccurs="*"/>
  </ElementType>
</Schema>
```

Linking Child Objects to Elements

XML objects do not support standard PowerDesigner link objects. To link a child object to an element, you must click the child object tool in the palette and then click the element symbol in the diagram. This will automatically create a link between both objects.

The following table lists the allowed links:

Tool	Action
	If you click a parent element symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet
	If you click the upper part of a child element symbol with the Element tool, a brother element symbol is displayed above the child element symbol 
	If you click the middle part of a child element symbol with the Element tool, a sequence group particle and a grand child element symbol are created. You can modify the group particle via its property sheet
	If you click the lower part of a child element symbol with the Element tool, a brother element symbol is displayed below the child element symbol
	If you click an element symbol with the *Any* tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet 
	If you click an element symbol with the Group tool, a referencing group is created. You must now select a group for the reference 
	If you click an element symbol with the Complex Type tool, a complex type symbol is displayed superposed, but not linked, to the element symbol. A global complex type cannot be the child of an element
	If you click an element symbol with the Sequence tool, a sequence group particle is displayed linked to the element symbol 

Tool	Action
	If you click an element symbol with the Choice tool, a choice group particle is displayed linked to the element symbol 
	If you click an element symbol with the *All* tool, an all group particle is displayed linked to the element symbol 

Note: When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See elements in Tool column).

Manipulating XML Objects Graphically

The graphical interface of PowerDesigner allows you to manipulate your XML schema by moving XML objects within the Browser, within the diagram, or by dragging them from the Browser into the diagram or from the diagram into the Browser.

Global Objects

Note: Select **Tools > General Options** to make sure that Move is the Default action of the Drag & Drop option.

A global object is right under the model item in the Browser tree view. It has no parent symbol in the diagram. You can move global objects within or between the Browser tree view and the diagram window:

- *Within Browser* - You can move a global object within the Browser to convert it into a local object (under a group particle item), but you cannot move a global object within its own structure (as a child of itself).
- *Within Diagram* - You can move a global object within the diagram to convert it into a local object. Just move the global object symbol to a group particle symbol.
- *From Browser to Diagram* - When you move a global object from the Browser to the diagram, a synonym symbol is created in the diagram.
- *From Diagram to Browser* - You can move a global object from the diagram to the Browser, and convert it into a local object (under a group particle item). If the new local object does not appear in the diagram, double-click the Collapse node of the group particle symbol under which the former global object has been attached.

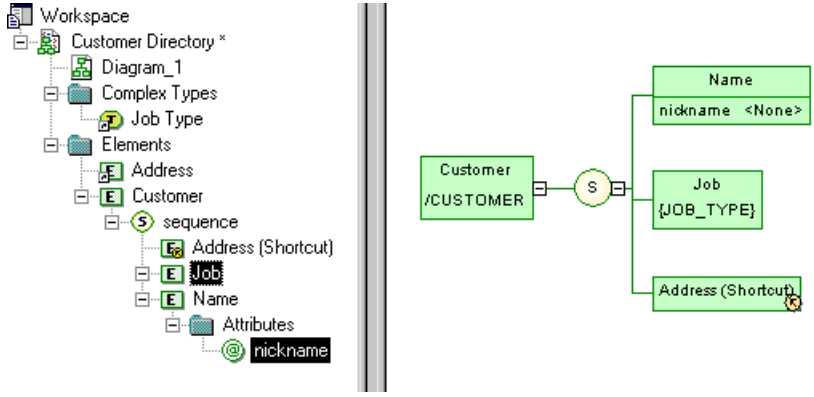
Local Objects

A local object is under a group particle item in the Browser tree view. It has a parent symbol in the diagram. You can move local objects within or between the Browser tree view and the diagram window:

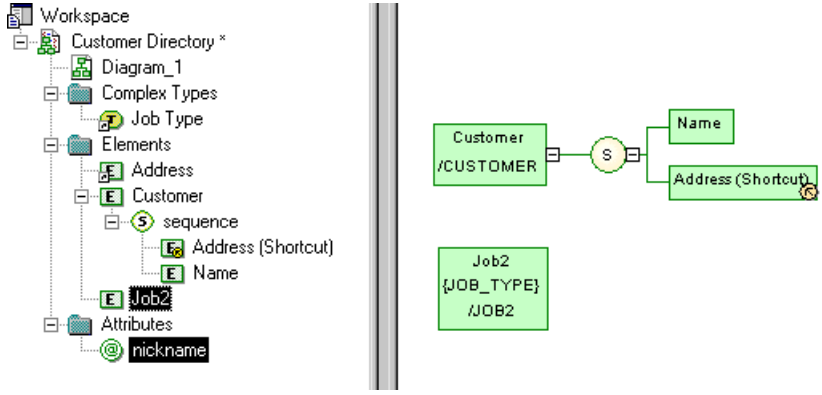
- *Within Browser* - You can move a local object within the Browser to convert it into a global object. If the new global object does not appear in the diagram, select **Symbol > Show Symbols** and click the corresponding tab to select the new global object
- *Within Diagram* - You can move a local object within the diagram to another group particle. It remains a local object, but with a new parent object. You cannot move a local object within the diagram to convert it into a global object. It remains attached to its group particle.
- *From Browser to Diagram* - When you move a local object from the Browser to the diagram, a synonym is created, attached to the same group particle as the original symbol.
- *From Diagram to Browser* - You can move a local object from the diagram to the Browser, and convert it into a global object. If the new global object does not appear in the diagram, select **Symbol > Show Symbols** and click the corresponding tab to select the new global object

Example: Converting a Local Object into a Global Object

In the following example, Job is a child element of the Customer element and Nickname is the attribute of the Name element:



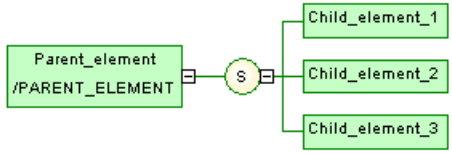
The Job entry in the Browser is selected and dragged and dropped onto the model element, Customer Directory and the Nickname element is similarly dragged and dropped onto the model. Job becomes Job2, a global element, and Nickname becomes a global attribute:



Group Particles (XSM)

An element composed of other elements is a parent element with child elements.

Child elements are linked to their parent element through a group particle.



There are three kinds of group particles:

Tool	Symbol	Description
		Sequence - Child elements must appear at least once in the order of their declaration
		Choice - Only one child element can be linked to the parent element
		All - Child elements can appear in any order and each of them once or not at all

These particles translate to the following tags in each of the supported languages:

Group Particle	XSD	XDR (order attribute)	DTD (separator)
Sequence	<sequence>	seq	, (comma)
Choice	<choice>	one	| (bar)

Group Particle	XSD	XDR (order attribute)	DTD (separator)
All	<all>	many	, (comma)

Creating a Group Particle

You can create a group particle in any of the following ways:

- Use the Sequence, Choice, or All tool in the diagram Palette.
- Open the property sheet of an element, group, or complex type, and select a group particle from the Group type list.

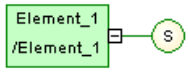
For general information about creating objects, see the "Objects chapter" in the *Core Features Guide*.

Creating a Group Particle from the Palette

You can create a group particle with the Group Particle tool in the palette.

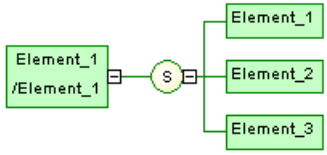
1. Select a group particle tool (Sequence, Choice, or All) in the palette, and then click the element symbol in the diagram.

   A group particle is created and its symbol is displayed in the diagram, linked to the element symbol.



2. Select the Element tool in the palette, and then click the group particle symbol to add an element item. Click the symbol again to add additional elements.

   The child elements appear one by one in the diagram, linked to the group particle symbol.



3. Right-click in order to recover the Pointer.

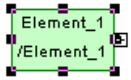
Note: When you click an element symbol with the Element tool, a sequence symbol (by default) is displayed in the diagram between the parent element and the child element. To add other child elements, click the sequence symbol with the Element tool. To change the group particle, double-click the sequence symbol to display its property sheet, then select another group particle in the Type list and click OK.

Creating a Group Particle from the Property Sheet of an Element

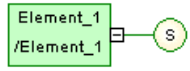
You can create a group particle from the property sheet of an element.

1. Open the property sheet of the element, select a group particle from the Group type list, and then click OK.

   The element is displayed selected, with an *Expand tab* (+) on its right side:

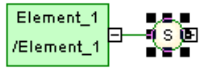
2. Click an empty space in the diagram to deselect the element, and then click the Expand tab (+) to reveal the group particle symbol (in this case, a sequence):

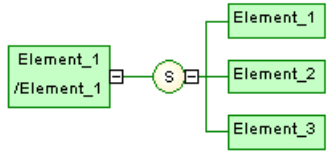
   Note that you can click the *Collapse tab* (-), on the link in order to hide the group particle.

3. Double-click the group particle to open its property sheet, and then click the Items tab.
4. Click the *Add Element* tool for each child element you want to create in the list.
5. Click OK to return to the diagram.

   The group particle is displayed selected, with an Expand tab on its right side.

6. Click an empty space in the diagram, to deselect the group particle symbol, and then click the Expand tab to reveal the child element symbols.

Note: Child elements are defined within the namespace of their parent element. Therefore, there cannot be a conflict between a parent and a child name.

For more information on the namespace concept, see "Controlling the namespace of a package", in the Models chapter of the *Core Features Guide*.

Group Particle Properties

You can modify an object's properties from its property sheet. To open a group particle property sheet, double-click its diagram symbol or its Browser entry beneath its parent object.

The following sections detail the property sheet tabs that contain the properties most commonly entered for group particles.

The General tab contains the following properties:

Property	Description
Type	Type of the group particle. You can change its type by selecting a value in the list and clicking OK
Minimum	Minimum number of times the group particle can occur. To specify that the group particle is optional, set this property to zero
Maximum	Maximum number of times the group particle can occur. For an unlimited number of times, select unbounded
ID	ID of the group particle. Its value must be of type ID and unique within the model containing the group particle

Items Tab

This tab list the child elements associated with the group particle. You can add additional children directly on this tab using the following tools:

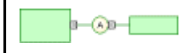
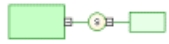
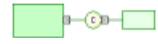
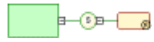
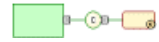
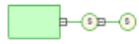
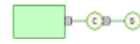
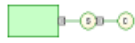
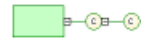
Tool	Description
	Add Element - Adds an element to the list
	Add Any - Adds an Any to the list. Only available with a choice or a sequence group particle
	Add Group Particle - Adds a group particle to the list
	Add Reference to Element - Adds a referencing element to the list. Select a global element for the reference in the Selection dialog box. To use this tool, you must have previously defined a global element in the current model
	Add Reference to Group - Adds a referencing group to the list. Select a group for the reference in the Selection dialog box. To use this tool, you must have previously defined a group in the current model

Adding a Child Object to a Group Particle

XML objects do not support standard link objects. To link a child object to a group particle, you must click the child object tool in the palette and then click the group particle symbol in the diagram. This will automatically create a link between both objects.

The following table lists the allowed links:

Warning! A group particle cannot be created from scratch in a diagram. It must be the child element of an element, a group or a complex type.

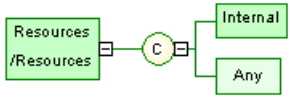
Tool	Sequence symbol	Choice symbol	All symbol
			
Any			No link
	 A referencing group is created. You must now select a group for the reference	 A referencing group is created. You must now select a group for the reference	No link
	No link	No link	No link
			No link
			No link
All	No link	No link	No link

Note: When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

Any Elements (XSM)

Any elements allow you to attach any type of object to a choice or a sequence group particle.

The following illustration shows an Any in a diagram:



- In an XSD file, Any is declared with the <any> tag:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="RESOURCES">
      <xs:complexType>
         <xs:choice>
            <xs:element name="INTERNAL"/>
            <xs:any namespace="##local" processContents="lax"/>
         </xs:choice>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

- In a DTD file, Any is declared within an <!ELEMENT> tag with the keyword "ANY":

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT Resources ANY>
<!ELEMENT Internal EMPTY>
```

- In an XDR file, Any is declared through of an <ElementType> tag (resources in the example) with its *model* attribute set to "*open*". Although it is displayed in a diagram, Any is not considered as an object in an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_Any"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="internal" content="empty"/>
  <ElementType name="resources" model="open" content="eltOnly" order="one">
     <element type="internal"/>
  </ElementType>
</Schema>
```

Creating an Any Element

You can create an any element in any of the following ways:

- Use the Any tool in the diagram Palette.
- Open the Items tab in the property sheet of a group particle, and click the Add Any tool.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Any Element Properties

You can modify an object's properties from its property sheet. To open an any element property sheet, double-click its diagram symbol or its Browser entry beneath its parent object.

The General tab contains the following properties:

Property	Description
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Minimum	Minimum number of times the Any can occur. To specify that the Any is optional, set this attribute to *zero*
Maximum	Maximum number of times the Any can occur. For an unlimited number of times, select *unbounded*
ID	ID of the Any. Its value must be of type ID and unique within the model containing the Any. Only available in a model targeted with XSD

Property	Description
Namespace	Namespaces containing the objects that can be used. If you select *##any*, objects from any namespace can be used. If you select *##other*, objects from any namespace other than the target namespace of the schema can be used. If you select *##local*, objects that are not qualified with a namespace can be used. If you select *##target-Namespace*, objects from the target namespace of the schema can be used. If you type a combination of URI references, ##targetNamespace and ##local, provided they are separated by a white space, objects from this combination can be used. Only available in a model targeted with XSD
Process contents	Indicator of how an XML processor should handle validation of XML documents containing the objects specified by the Any. If you select *Strict*, the XML processor must obtain the schema and validate any object of the specified namespaces. If you select *Lax*, the XML processor will try to obtain the schema and validate any object of the specified namespaces. If the schema cannot be found, no error will occur. If you select *Skip*, the XML processor will not try to validate the objects of the specified namespaces. Only available in a model targeted with XSD

Attributes (XSM)

Attributes are used to give additional information about elements.

There are global and local attributes:

- Global attributes are defined with the Model menu. In a schema, they are directly linked to the <schema> tag (root element). They can be reused for any element in the model through references (See "NUMBER" attribute in the generated schema)
- Local attributes only apply to the elements in which they are created. They can be defined by reference to a global attribute (See Reference property)

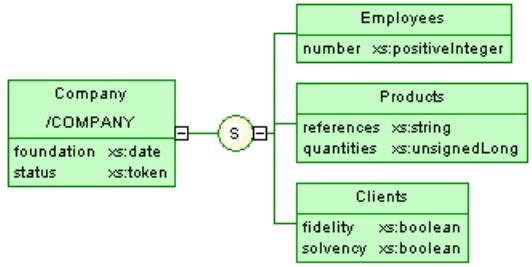
Note: In a model targeted with the XML-Data Reduced language, local attributes are first declared separately, like global attributes (with the <AttributeType> tag and a name attribute), then within their parent element (with the <attribute> tag and a type attribute).

Extract of an XDR file:

```
<AttributeType name="globalAttribute"/>
<ElementType name="parentElement" content="empty">
   <AttributeType name="localAttribute" default="0"
   <attribute default="0" type="localAttribute"/>
</ElementType>
```

You can derive an attribute data type to extend or restrict its values. (Only with a model targeted with XSD)

For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="COMPANY">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="EMPLOYEES">
               <xs:complexType>
                  <xs:attribute ref="NUMBER">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="PRODUCTS">
               <xs:complexType>
                  <xs:attribute name="REFERENCES" type="xs:string">
                  </xs:attribute>
                  <xs:attribute name="QUANTITIES" type="xs:unsignedLong">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="CLIENTS">
               <xs:complexType>
                  <xs:attribute name="FIDELITY" type="xs:string">
                  </xs:attribute>
                  <xs:attribute name="SOLVENCY" type="xs:string">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="FOUNDATION" type="xs:date">
         </xs:attribute>
         <xs:attribute name="STATUS" type="xs:token">
         </xs:attribute>
      </xs:complexType>
   </xs:element>
   <xs:attribute name="NUMBER" type="xs:positiveInteger">
   </xs:attribute>
</xs:schema>
```

In a schema, attributes are declared with <attribute> tags.

Creating an Attribute

You can create attributes (and attribute groups) on the Attributes tab of an element, complex type, or attribute group using the following tools:

Tool	Description
	Add Attribute - Creates a local attribute
	Add Undefined Reference to Attribute Group - Adds an attribute group with a reference to an attribute group defined in the current model. Select a name in the Reference list. You can also type a new name in the Reference column and then define a new attribute group in the Attribute Groups list (See Model menu)
	Add Reference to Attribute - Adds one or several attributes with a reference to global attributes defined in the current model. Select one or several global attributes in the Selection dialog box
	Add Reference to Attribute Group - Adds a reference to an attribute group defined in the current model. Select one or several attribute groups in the Selection dialog box
	Any Attribute - Adds "any" attribute of a specified namespace. For more information, see *"Any" Attributes* on page 40.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Attribute Properties

You can modify an object's properties from its property sheet. To open an attribute property sheet, double-click its diagram symbol or its Browser entry in the Attributes folder beneath an entity or complex type.

The General tab contains the following properties:

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users.
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces.
Comment	Descriptive label of the attribute.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.

Property	Description
Reference	Name of an attribute in the current model or another model opened in the workspace. A reference allows you to reuse an attribute with all its properties without having to define it again. Use the list to select an attribute in the current model. Use the Browse tool to select an attribute from any model opened in the workspace. If you select an attribute from another model, a shortcut is created with the referencing attribute. When you define a reference, the name and code are those of the target attribute and these properties are grayed.
Type	Attribute data type, which must be a qualified name. Use the list to select a built-in data type. Use the Browse tool to select a simple type defined in the current model or another model opened in the workspace.
Embedded Type	If selected, the attribute data type disappears and a <simple type> tag is created in the schema within the <attribute> tag. Only available in a model targeted with XSD.
Derivation	Derivation method for the attribute data type. Used to extend or restrict the values of the attribute data type. When you define a derivation, the data type disappears. You must click Apply and then the Properties tool to select a type, a base type or member types for the corresponding derivation (list, restriction or union). Only available in a model targeted with XSD.

Detail Tab

The Detail tab of an attribute property sheet displays the following properties:

Property	Description
Default	Default value. Enter a default value only if there is no fixed value.
Fixed	Fixed value. Enter a fixed value only if there is no default value.
Use	Indicator of how the attribute is used. If you select *Optional*, the attribute is optional and may have any value. If you select *Prohibited*, the attribute cannot be used. Use this value to prohibit the use of an existing attribute in the restriction of another complex type. If you select *Required*, the attribute must appear at least once and may have any value matching its data type.
Form	Form of the attribute. If you select *Qualified*, Form must be qualified by combining the target namespace of the schema with the no-colon-name of the attribute. If you select *Unqualified*, Form is not required to be qualified with the namespace prefix and is matched against the no-colon-name of the attribute.
ID	ID of the attribute. Its value must be of type ID and unique within the model containing the attribute.

Attribute Property Sheet Values Tab

The Values tab of an attribute property sheet is only available in a model targeted with DTD or XDR. You can set a list of predefined values for an attribute.

Note: In a model targeted with the XML-Data Reduced language, there is also a Values tab in the element property sheet.

Defining Attributes in XDR Files

In an XML-Data Reduced language model, attributes tags are defined by different attributes:

XDR attribute attribute	Description
name	Specifies the name of the attribute. Tab: General Field: Name
default	Specifies a default value for the attribute. Tab: Detail Field: Default
dt:type	Specifies a type for the attribute. Tab: General Field: Type
dt:values	To specify a list of available values for a global attribute Tab: General Field: Values
type	Specifies the name of a global attribute as a reference for a local attribute. Tab: General Field: Reference

The following example shows an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_attributes"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <AttributeType name="globalAttribute"/>
  <ElementType name="parentElement" content="empty">
     <AttributeType name="localAttribute" default="0" dt:values="1 2 3"/>
     <attribute default="0" type="localAttribute"/>
  </ElementType>
</Schema>
```

Any Attributes

The "Any" attribute feature allows you to insert any attribute of specified namespaces into an element, a complex type or an attribute group declaration. It is only available in a model targeted with XSD.

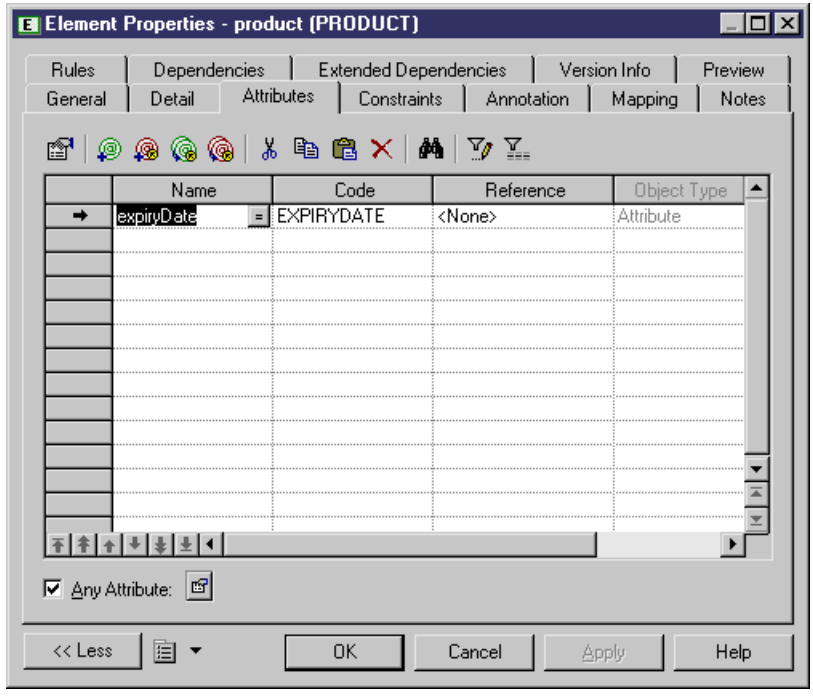
In a schema, Any Attribute is declared with the <anyAttribute> tag.

For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="PRODUCT">
      <xs:complexType>
         <xs:attribute name="EXPIRYDATE" type="xs:date">
         </xs:attribute>
         <xs:anyAttribute namespace="##local" processContents="skip"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Warning! Any Attribute only is displayed in a schema (see the Preview tab of a model property sheet).

The Any Attribute feature is available via a check box in the bottom-left corner of an Attributes tab.



To display an Any Attribute property sheet, select the Any Attribute check box and then click the Properties tool.

Any Attribute Property Sheet General Tab

The General tab contains the following properties:

Property	Description
ID	ID of the Any Attribute. Its value must be of type ID and unique within the model containing the Any Attribute
Namespace	Namespaces containing the attributes that can be used. If you select *##any*, attributes from any namespace can be used. If you select *##other*, attributes from any namespace other than the target namespace of the schema can be used. If you select *##local*, attributes that are not qualified with a namespace can be used. If you select *##targetNamespace*, attributes from the target namespace of the schema can be used. If you type a white space delimited list with URI references, ##targetNamespace and ##local, attributes from this list can be used
Process contents	Indicator of how an XML processor should handle validation of XML documents containing the attributes specified by the Any Attribute. If you select *Lax*, the XML processor will try to obtain the schema and validate any attribute of the specified namespaces. If the schema cannot be found, no error will occur. If you select *Skip*, the XML processor will not try to validate the attributes of the specified namespaces. If you select *Strict*, the XML processor must obtain the schema and validate any attribute of the specified namespaces

Constraints: Keys, Uniques, and KeyRefs (XSM)

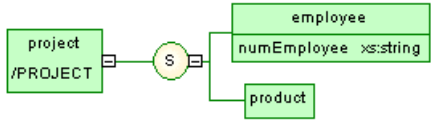
Constraints indicate that element values must be unique within their specified scope. Constraints are only available in a model targeted with XSD.

Each constraint has two specific attributes: selector and field.

In a schema, a constraint is declared with its corresponding tag: <unique>, <key> or <keyRef>.

There are three kinds of identity constraints:

- A *Unique* constraint - specifies that an element or an attribute value (or set of values) must be unique or null within a specified scope. For example:

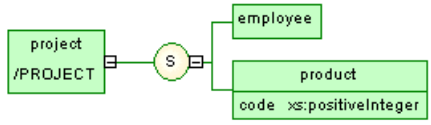


Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="PROJECT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="EMPLOYEE">
               <xs:complexType>
                  <xs:attribute name="NUMEMPLOYEE" type="xs:string">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="PRODUCT"/>
         </xs:sequence>
      </xs:complexType>
      <xs:unique name="UNIQUENUM">
         <xs:selector xpath="employee"/>
         <xs:field xpath="@numEmployee"/>
      </xs:unique>
   </xs:element>
</xs:schema>
```

The UNIQUENUM unique constraint, defined on the project element, specifies that the numEmployee attribute must be unique or null within the employee element

- A *Key* constraint - specifies that an element or an attribute value (or set of values) must be a key within a specified scope; the data must be unique, not null, and always present within a specified scope. For example:

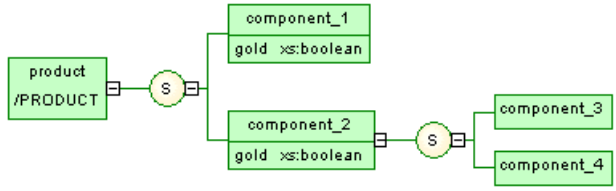


Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="PROJECT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="EMPLOYEE"/>
            <xs:element name="PRODUCT">
               <xs:complexType>
                  <xs:attribute name="CODE" type="xs:positiveInteger">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
      <xs:key name="KEYCODE">
         <xs:selector xpath="product"/>
         <xs:field xpath="@code"/>
      </xs:key>
   </xs:element>
</xs:schema>
```

The KEYCODE key constraint, defined on the project element, specifies that the code attribute must be unique, not null and always present within the product element.

- A *KeyRef* constraint - specifies that an element or attribute value (or set of values) corresponds to the value of a specified key or unique constraint. A keyRef is a reference to a key or a unique constraint. For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:k="keyRef.namespace">
   <xs:element name="PRODUCT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="COMPONENT_1">
               <xs:complexType>
                  <xs:attribute name="GOLD" type="xs:boolean">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="COMPONENT_2">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="COMPONENT_3"/>
                     <xs:element name="COMPONENT_4"/>
                  </xs:sequence>
                  <xs:attribute name="GOLD" type="xs:boolean">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
      <xs:unique name="UNIGOLD">
         <xs:selector xpath="k:COMPONENT_1"/>
         <xs:field xpath="@GOLD"/>
      </xs:unique>
      <xs:keyref name="KEYREF_UNIGOLD" refer="k:UNIGOLD">
         <xs:selector xpath="k:COMPONENT_2"/>
         <xs:field xpath="@GOLD"/>
      </xs:keyref>
   </xs:element>
</xs:schema>
```

The KEYREF_UNIGOLD keyRef, defined on the product element, by reference to the UNIGOLD unique constraint, specifies that the gold attribute must be unique or null within the component_2 element, as well as it must be unique or null within the component_1 element (See UNIGOLD).

Creating a Constraint

You create constraints on the Constraints tab of an element, using one of the following tools:

Tool	Description
	Add Key Constraint - The element value must be a key within the specified scope. The scope of a key is the containing element in an instance document. A key must be unique, not null, and always present

Tool	Description
	Add Unique Constraint - The element value must be unique or null within the specified scope
	Add KeyRef Constraint - The element value corresponds to those of the specified key or unique constraint

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Constraint Properties

You can modify an object's properties from its property sheet. To open a constraint property sheet, double-click its diagram symbol or its Browser entry in the Constraints folder beneath an entity.

The General tab contains the following properties:

Property	Description
Name	Name of the constraint, which must be a no-colon-name.
Code	Code of the constraint, which must be a no-colon-name.
Comment	Descriptive label of the constraint.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.
ID	ID of the constraint. Its value must be of type ID and unique within the model containing the constraint.
Reference [KeyRef only]	Name of a key or a unique constraint defined in the current model (or another model with a specified namespace). The Reference value must be a qualified name.
Selector (XPath)	An XML Path Language expression that selects a set of elements across which the values specified in the Fields tab must be unique. There can only be one selector (see *Specifying a Constraint Selector* on page 45)

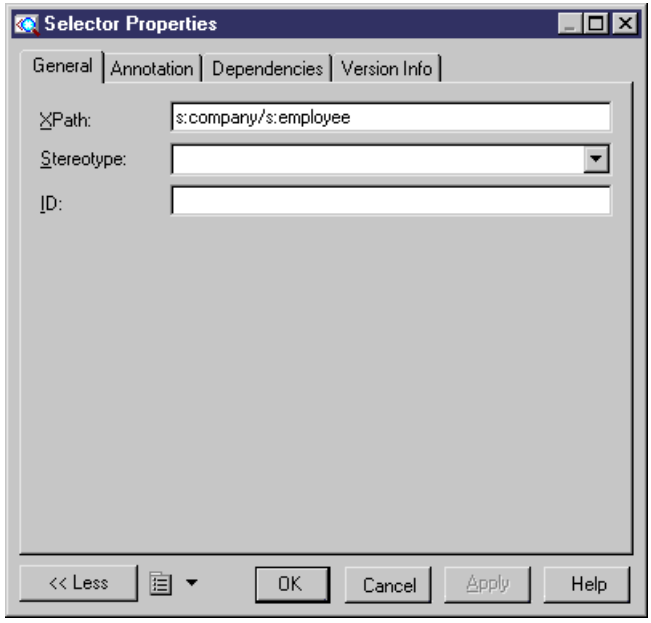
Fields Tab

The Fields tab lists XPath expressions used to define the constraint (see *XPath Abbreviated Syntax* on page 46). If more than one field is listed, the combination of fields must be unique (see *Specifying Constraint Fields* on page 45).

Specifying a Constraint Selector

A constraint selector specifies an XPath expression that selects a set of elements for a constraint.

1. Open the property sheet of a constraint, and enter an XPath expression in the Selector (XPath) field (see *XPath abbreviated syntax* on page 46).
2. Click Apply to make available the Properties tool to the right of the field.
3. Click the Properties tool to open the selector property sheet.

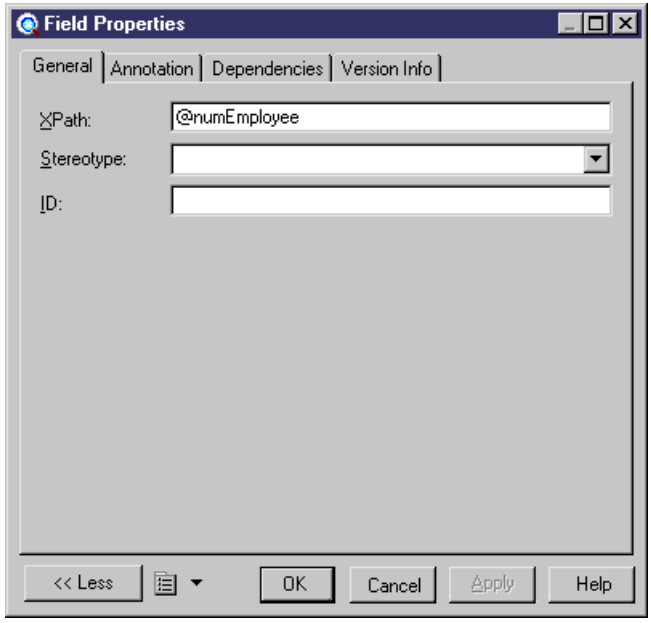


4. [optional] Enter additional properties for the selector, and then click OK to return to the constraint.

Specifying Constraint Fields

Constraint fields are XPath expressions used to define a constraint (unique, key or keyRef).

1. Open the property sheet of a constraint and then click the Fields tab.
2. Click in the XPath column and enter an XPath expression (see *XPath abbreviated syntax* on page 46).
3. Click the Properties tool to open the property sheet of the field:



4. [optional] Enter additional properties for the selector, and then click OK to return to the constraint.

XPath Abbreviated Syntax

An XPath expression allows you to locate a node (an element with its ramifications) in the hierarchical tree structure of an XML document.

The XPath expressions permitted to define constraint selectors and fields are limited to a subset of the full XPath language defined in the W3C Recommendation XML Path Language 1.0:

Syntax	Description
/	Root node of the XML document. It is the root element with its ramifications
.	Selects the context node. It is the current element (on which an identity constraint is defined) with its ramifications
..	Selects the context node parent
*	Selects all the child elements of the context node
employee	Selects all the employee child elements of the context node
s:employee	Selects all the employee child elements of the context node, defined in the namespace with the "s" prefix

Syntax	Description
@numEmployee	Selects the numEmployee attribute of the context node
@*	Selects all the attributes of the context node
../@numEmployee	Selects the numEmployee attribute of the context node parent
employee[1]	Selects the first employee child element of the context node
employee[last()]	Selects the last employee child element of the context node
*/employee	Selects all the employee grandchildren of the context node
//employee	Selects all the employee descendants of the root node
.//employee	Selects the employee descendants of the context node
company//employee	Selects the employee descendants of the company child elements of the context node
//company/employee	Selects all the employee elements with company as parent element in the context node
/book/chapter[2]/section[3]	Selects the third section in the second chapter of the book
employee[@dept="doc"]	Selects all the employee child elements of the context node with a dept attribute set to doc
employee[@dept="doc"][3]	Selects the third employee child element of the context node with a dept attribute set to doc
employee[3][@dept="doc"]	Selects the third employee child element of the context node only if it has a dept attribute set to doc
chapter[title]	Selects the chapter child elements of the context node with at least one title child element
chapter[title="About this book"]	Selects the chapter child elements of the context node with at least one title child element with a text content set to About this book
employee[@numEmployee and @dept]	Selects all the employee child elements of the context node with the numEmployee and dept attributes
text()	Selects all the child nodes of the text context node

Selector and Field Property Sheet General Tab

The General tab of a selector or field property sheet contains the following properties:

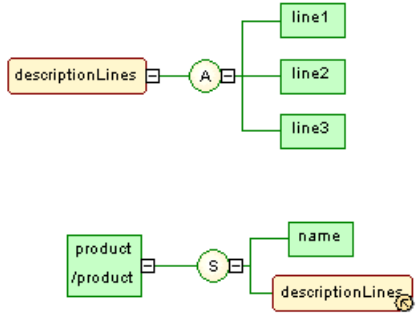
Property	Description
XPath	For a selector: An XPath expression relative to the parent element being declared. It identifies the child elements to which the identity applies For a field: An XPath expression relative to each element selected by the selector of the constraint. It identifies a single element (with a simple type) whose content or value is used for the constraint
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the selector. Its value must be of type ID and unique within the model containing the selector

Groups (XSM)

A group of elements is a set of elements arranged by a group particle (all, choice or sequence), which is then referenced in the model by various elements.

- A *group* - is created independently, without a parent element, and can be reused multiple times by elements, complex types or other global groups, through references. In a schema, it is directly linked to the <schema> tag (root element). See *Creating a group* on page 50.
- A *reference to a group* - is created within an element, complex type or global group, and makes the referenced group available to its parent. See *Creating a reference to a group* on page 50.

For example:



The descriptionLines group is reused in the definition of the product element by clicking the sequence group particle (S) with the palette Group tool. The Reference property of the referencing group property sheet is then set to descriptionLines.

In the generated XSD file, the group is first declared with the <group> tag and then reused through a reference (ref) set to descriptionLines:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:group name="descriptionLines">
      <xs:all>
         <xs:element name="line1"/>
         <xs:element name="line2"/>
         <xs:element name="line3"/>
      </xs:all>
   </xs:group>
   <xs:element name="product">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="name"/>
            <xs:group ref="descriptionLines">
            </xs:group>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

- In the generated DTD file, the group is expanded directly within its parent element:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT line1 EMPTY>
<!ELEMENT line2 EMPTY>
<!ELEMENT line3 EMPTY>
<!ELEMENT product (name,line1,line2,line3)>
<!ELEMENT name EMPTY>
```

- In the generated XDR file, the group is declared through a <group> tag, within an <ElementType> tag with its order attribute set to seq:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_group"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="line1" content="empty"/>
  <ElementType name="line2" content="empty"/>
  <ElementType name="line3" content="empty"/>
  <ElementType name="name" content="empty"/>
  <ElementType name="product" model="closed" content="eltOnly" order="seq">
     <element type="name"/>
     <group>
        <element type="line1"/>
        <element type="line2"/>
        <element type="line3"/>
     </group>
  </ElementType>
</Schema>
```

Note: In a model targeted with DTD or XDR language, there are no global or referencing groups, although they appear on the diagram. Groups are expanded within their parent element and their child elements are declared individually as global elements. (See generated DTD and XDR files in *Groups (XSM)* on page 48)

Creating a Group

A group is created independently in the diagram, and will be reused within other elements by way of references.

For more information on references, see *Creating a reference to a group* on page 50.

You can create a group in any of the following ways:

- Select the Group tool in the diagram Palette and click in an empty space in the diagram.
- Select **Model > Groups** to access the List of Groups, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select **New > Group**.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Creating a Reference to a Group

A reference to a group is created as a child of an element, group or complex type, and makes the referenced group available to its parent.

You can create a referencing group in any of the following ways:

- Select the Group tool in the diagram Palette, and click on an element, group, or complex type symbol.
- On the Items tab of the property sheet of a group particle, click the Add Group with Reference to Group tool (see *Group Particle Properties* on page 31).

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Group Properties

You can modify an object's properties from its property sheet. To open a group (or reference to a group) property sheet, double-click its diagram symbol or its Browser entry in the Groups folder (or, in the case of a reference to a group, beneath its parent object).

The General tab contains the following properties:

Property	Description
Name	[unavailable to references to groups] Name of the group, which must be a no-colon-name. Required when the group is global
Code	[unavailable to references to groups] Code of the group, which must be a no-colon-name. Required when the group is global
Comment	Descriptive label for the group.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.

Property	Description
Reference	[for references to groups only] Name of a group in the current model or another model opened in the workspace, which must be a qualified name. A reference allows you to reuse a group with all its properties without having to define it again. Use the list to select a group in the current model. Use the **Browse** tool to select a group from any model opened in the workspace. If you select a group from another model, a shortcut is created with the referencing group. When a reference is defined, the name and code are those of the target group, and the properties are grayed. Once you have defined the reference of a referencing group, you can locate the referenced group in the diagram by right-clicking the referencing group symbol and selecting Find Referenced Group in the contextual menu. The referenced group is displayed with handles in the diagram.
Group type	[unavailable to references to groups] Specifies how child elements are to be used within the group. You can choose between: • all • choice • sequence For more information, see *Group Particles (XSM)* on page 29.
Minimum	Minimum number of times the group can occur. To specify that the group is optional, set this attribute to *zero*.
Maximum	Maximum number of times the group can occur. For an unlimited number of times, select *unbounded*.

You can access the **Preview** tab of a group property sheet directly, by right-clicking the group symbol in the diagram and selecting **Preview**.

Linking Child Objects to a Group

XML objects do not support standard link objects. To link a child object to a group, you must click the child object tool in the palette and then click the group symbol in the diagram. This will automatically create a link between both objects.

The following table lists the allowed links:

Tool	Action
	If you click a group symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet
	If you click a group symbol with the *Any* tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet

Tool	Action
	If you click a group symbol with the Group tool, a sequence group particle and a referencing group are created. You can modify the group particle via its property sheet. You must now select a group for the reference
	If you click a group symbol with the Complex Type tool, a complex type symbol is displayed superposed, but not linked, to the group symbol. A global complex type cannot be the child of a group
	If you click a group symbol with the Sequence tool, a sequence group particle is displayed linked to the group symbol
	If you click a group symbol with the Choice tool, a choice group particle is displayed linked to the group symbol
	If you click a group symbol with the *All* tool, an all group particle is displayed linked to the group symbol

Note: When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

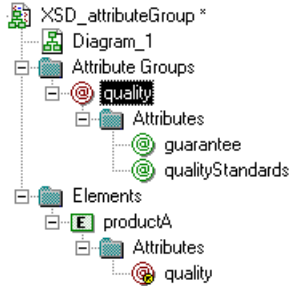
Attribute Groups (XSM)

Attribute groups are not supported by XDR.

An attribute group is a set of attributes, which is referenced in the model by various elements:

- An *attribute group* - is created independently, without a parent element, and can be reused multiple times by elements, complex types or other global attribute groups, through references. In a schema, it is directly linked to the <schema> tag (root element). See *Creating an attribute group* on page 53.
- A *reference to an attribute group* - is created within an element, complex type, or global attribute group, and makes the referenced attribute group available to its parent. See *Creating a reference to an attribute group* on page 54.

For example:



The quality attribute group is composed of the guarantee and qualityStandards attributes. The productA element reuses the quality attribute group via the Attributes tab of its property sheet.

- Generated XSD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:attributeGroup name="quality">
      <xs:attribute name="guarantee">
      </xs:attribute>
      <xs:attribute name="qualityStandards">
      </xs:attribute>
   </xs:attributeGroup>
   <xs:element name="productA">
      <xs:complexType>
         <xs:attributeGroup ref="quality">
         </xs:attributeGroup>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

  In a schema, a group of attributes is declared with the <attributeGroup> tag. It can contain the following tags: <attribute>, <attributeGroup> or <anyAttribute>.

- Generated DTD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT productA EMPTY>
<!ATTLIST productA
          guarantee            CDATA
          qualityStandards     CDATA>
```

Creating an Attribute Group

An attribute group is created independently, and will be reused within other elements by way of references.

For more information on references, see *Creating a reference to an attribute group* on page 54.

You can create an attribute group in any of the following ways:

- Select **Model > Attribute Groups** to access the List of Attribute Groups, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select **New > Attribute Group**.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Creating a Reference to an Attribute Group

A reference to an attribute group is created within an element, complex type, or attribute group, and makes the referenced attribute group available to its parent. You can create a reference to an attribute group as follows:

You can create an attribute group in the following way:

- On the Attributes tab of the property sheet of an element, complex type, or attribute group, click the Add Group with Reference to Group tool (see *Element Properties* on page 21).

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Attribute Group Properties

You can modify an object's properties from its property sheet. To open an attribute group (or reference to an attribute group) property sheet, double-click its Browser entry in the Attribute Groups folder (or, in the case of a reference to an attribute group, beneath its parent object). The following sections detail the property sheet tabs that contain the properties most commonly entered for attribute groups.

The General tab contains the following properties:

Property	Description
Name	[unavailable to references to attribute groups] Name of the attribute group, which must be a no-colon-name. Required when the attribute group is global.
Code	[unavailable to references to attribute groups] Code of the attribute group, which must be a no-colon-name. Required when the attribute group is global.
Comment	Descriptive label of the attribute group.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.
Reference	[for references to attribute groups only] Name of an attribute group from the current model or any model opened in the workspace, which must be a qualified name. If you select an attribute group from another model, a shortcut is created with the referencing attribute group. A reference allows you to reuse an attribute group with all its properties without having to define it again. When a reference is defined, the name and code properties are grayed. The name and code are those of the target attribute group
ID	ID of the attribute group. Its value must be of type ID and unique within the model containing this attribute group

Attributes Tab
This tab lists the attributes and attribute groups associated with the attribute group. For information about the tools available on this tab for adding attributes and attribute groups, see *Creating an Attribute* on page 37.

Simple Types (XSM)

You can only create simple types in a model targeted with XSD.

A simple type is a data type definition for elements or attributes with text-only content. It cannot contain elements or attributes.

A simple type is defined by derivation of an existing simple type (built-in data type or derived simple type).

There are three kinds of derivation for a simple type:

- List - contains a white space-separated list of values of an inherited simple type
- Restriction - has a range of values restricted to a subset of those of an inherited simple type
- Union - contains a union of values of two or more inherited simple types

For more information on simple type derivations, see *Derivations: Extensions, Restrictions, Lists and Unions (XSM)* on page 61.

Once defined in a model, a simple type can be reused in the definition of an attribute, an element or a complex type.

Example of a simple type in a schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:simpleType name="BARCODE">
     <xs:restriction base="xs:nonNegativeInteger" id="STR1">
        <xs:length value="13"/>
     </xs:restriction>
   </xs:simpleType>
   <xs:element name="PRODUCTA" type="BARCODE"/>
   <xs:element name="PRODUCTB" type="BARCODE"/>
</xs:schema>
```

Creating a Simple Type

You can create a simple type in any of the following ways:

- Select **Model > Simple Types** to access the List of Simple Types, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select **New > Simple Type**.

Warning! If the simple type symbol does not appear in the diagram, select **Symbol > Show Symbols**, then click the Simple Type tab and select the simple type box to display its symbol in the diagram.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Simple Type Properties

You can modify an object's properties from its property sheet. To open a simple type property sheet, double-click its diagram symbol or its Browser entry in the Simple Types folder.

The General tab contains the following properties:

Property	Description
Name	Name of the simple type, which must be a no-colon-name. If specified, it must be unique among all simple types and complex types
Code	Code of the simple type, which must be a no-colon-name. If specified, it must be unique among all simple types and complex types
Comment	Descriptive label of the simple type.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.
Derivation	Derivation method for the simple type. Enabled and required when the simple type is defined.
Final	Property to prevent derivation of the current simple type.
ID	ID of the simple type. Its value must be of type ID and unique within the model containing the simple type.

Complex Types (XSM)

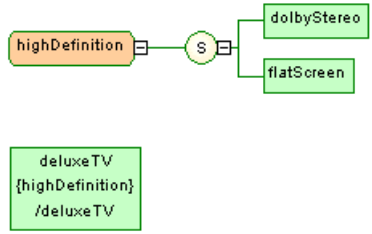
You can only create complex types in a model targeted with XSD.

A complex type is a data type definition used to define attributes and child elements of a parent element. It is a template for a data type definition that can be reused and derived by extension or restriction.

A complex type has:

- a global scope when it has no parent element in the diagram and when it is directly linked to the <schema> tag. It can then be reused or derived, by extension or restriction, in other parts of the schema.
- a local scope when integrated into an <element> tag. It applies only to its containing element.

The following illustration shows a diagram containing a complex type:



In the example above, HighDefinition is a global complex type, reused as data type for the deluxeTV element.

The generated schema is the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:complexType name="highDefinition">
      <xs:sequence>
         <xs:element name="dolbyStereo"/>
         <xs:element name="flatScreen"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="deluxeTV" type="highDefinition"/>
</xs:schema>
```

Warning! Global complex types appear in the model as objects, with their corresponding symbol in the diagram. Local complex types only appear in the schema (see Preview tab of an element property sheet).

Creating a Complex Type

You can create a complex type in any of the following ways:

- Use the Complex Type tool in the diagram Palette.
- Select **Model > Complex Types** to access the List of Complex Types, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select **New > Complex Type**.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Complex Type Properties

You can modify an object's properties from its property sheet. To open a complex type property sheet, double-click its diagram symbol or its Browser entry in the Complex Types folder. The following sections detail the property sheet tabs that contain the properties most commonly entered for complex types.

The General tab contains the following properties:

Property	Description
Name	Name of the complex type, which must be a no-colon-name and unique among all simple types and complex types.
Code	Code of the complex type, which must be a no-colon-name and unique among all simple types and complex types.
Comment	Descriptive label of the complex type.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.
Group type	Specifies that the complex type has child elements, and how they are used (see *Group Particles (XSM)* on page 29). You can choose between: • all – All children may be present. • choice – Only one child must be present. • group – Reference to a predefined group (see *Groups (XSM)* on page 48) • sequence – All children must be present in order.
Content	Content type of the complex type (see *Specifying the Type of Content of a Complex Type* on page 60).
Derivation	Derivation method for the complex type. Once you have selected a derivation method, you must define a base type. Click the Properties tool beside the derivation box to display the derivation property sheet. In the General tab, select a base type in the Base Type list.

Detail Tab

The Detail tab contains the following properties:

Property	Description
Final	Property to prevent derivation of the current complex type
Block	Property to prevent another complex type with the specified type of derivation from being used in place of the current complex type
Mixed	If selected, this property indicates that character data is allowed to appear between the child elements of the current complex type. Select Mixed only if the current complex type has a complex content (See general properties)
Abstract	If selected, this property indicates that the complex type can be used in the instance document
ID	ID of the complex type. Its value must be of type ID and unique within the model containing this complex type

Mapping Tab

This tab lets you map the complex type to PDM or OOM objects.

You associate one or more PDM abstract data types or OOM classes to the complex type using the **Add Objects** tool on the **Complex Type Sources** sub-tab. You can associate PDM abstract data type attributes or OOM class attributes to the complex type attributes using following tools on the **Attributes Mapping** tab:

Tool	Description
	Add Mapping - Selects the attributes in the current complex type that will be mapped to PDM abstract data type attributes or OOM class attributes. Once you have selected the attributes, you can use the list in the Mapped to column to select corresponding PDM abstract data type attributes or OOM class attributes
	Create from Sources - Copies PDM abstract data type attributes or OOM class attributes in the data source to the current complex type attributes
	Generate Mapping - Automatically generates a mapping between PDM abstract data type attributes or OOM class attributes and complex type attributes with the same name or code in the data source and the current model

For detailed information about mappings, see *Chapter 12, Object Mappings* in the *Core Features Guide* .

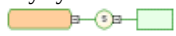
The following tabs are also available:

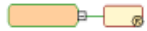
- Attributes - lists the attributes and attribute groups associated with the complex type (see *Attributes (XSM)* on page 35).

Linking a Child Object to a Complex Type

XML objects do not support standard link objects. To link a child object to a complex type, you must click the child object tool in the palette and then click the complex type symbol in the diagram. This will automatically create a link between both objects.

The following table lists the allowed links:

Tool	Action
	If you click a complex type symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet
	If you click a complex type symbol with the *Any* tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet 

Tool	Action
	If you click a complex type symbol with the Group tool, a referencing group is created. You can modify the group particle via its property sheet. You must now select a group for the reference 
	If you click a complex type symbol with the Complex Type tool, a second complex type symbol is displayed superposed, but not linked, to the first complex type symbol. A complex type cannot be the child of another complex type
	If you click a complex type symbol with the Sequence tool, a sequence group particle is displayed linked to the complex type symbol
	If you click a complex type symbol with the Choice tool, a choice group particle is displayed linked to the complex type symbol
	If you click a complex type symbol with the *All* tool, an all group particle is displayed linked to the complex type symbol

Note: When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

Specifying the Type of Content of a Complex Type

A complex type can contain either:

1. Open the property sheet of a complex type and select a type in the **Content** field:
   - simple content – character data or a simple type (but no elements), or
   - complex content – elements or elements and character data
2. Click **Apply** to make available the **Properties** tool to the right of the list.
3. [optional] Click the **Properties** tool to open the property sheet of the content, and specify an ID and, in the case of complex content, whether the content can be mixed:

Property	Description
ID	ID for the complex content. Its value must be of type ID and unique within the model containing the complex content
Mixed [complex content only]	Specifies that character data is allowed to appear between child elements of the complex type.

4. Click **OK** to return to the diagram.

Derivations: Extensions, Restrictions, Lists and Unions (XSM)

You can use derivations to extend or restrict the values of elements and of simple and complex types.

An XML model allows you to derive:

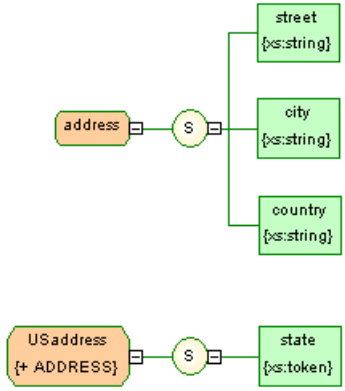
- Elements by extension, restriction, list or union
- Simple types by restriction, list or union
- Complex types by extension or restriction

Note: When you define a derivation in an element property sheet, a simple or a complex type is automatically created within the element declaration (See Preview tab). The Embedded type property is automatically set to Simple or Complex, and the Content property to Simple or Complex in the case of an embedded complex type.

Deriving by Extension

You can derive an element or complex type by extension to extend the values of its base type.

For example:



USaddress is a derivation by extension of the address complex type.

The generated schema is the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:complexType name="ADDRESS">
      <xs:sequence>
         <xs:element name="STREET" type="xs:string"/>
         <xs:element name="CITY" type="xs:string"/>
         <xs:element name="COUNTRY" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="USADDRESS">
      <xs:complexContent>
         <xs:extension base="ADDRESS">
           <xs:sequence>
              <xs:element name="STATE" type="xs:token"/>
           </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

1. Open the property sheet of an element or complex type and select Extension in the Derivation list.

   The Content field (and, in the case of an element, the Embedded type field) is set to Complex.
2. Click the Properties tool to the right of the Derivation box to open the property sheet of the extension and complete the following properties:

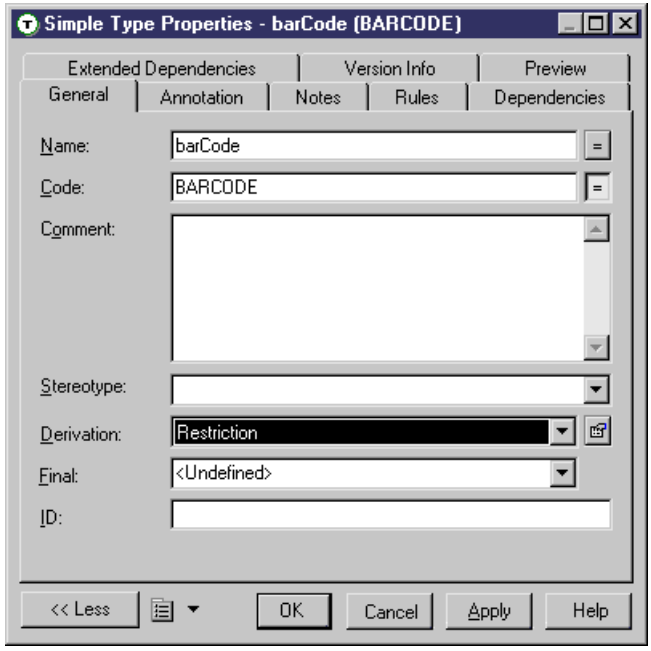
Property	Description
ID	ID of the extension. Its value must be of type ID and unique within the model containing the extension
Base Type	Data type on which the extension is based

3. Specify an ID, select a base type, and then click OK to return to the element or complex type.

Deriving by Restriction

You can derive an element, simple type, or complex type by restriction to restrict the values of their base type.

1. Open the property sheet of an element, simple type, or complex type, and select Restriction in the Derivation list.



For elements and complex types, the Content field (and, in the case of an element, the Embedded type field) is set to Complex.

2. Click the **Properties** tool to the right of the **Derivation** field to open the restriction property sheet, and complete the following fields on the **General** tab:

Property	Description
ID	ID of the simple type restriction. Its value must be of type ID and unique within the model containing the simple type restriction
Base type	Data type on which the restriction is based. Select a data type in the Base type list or with the Browse tool
Embedded type [simple types only]	If selected, the base type disappears and a simple type is created in the schema within the current simple type. Click Apply, and then the Properties tool beside the Embedded type box, to define a derivation and a base type for the embedded simple type.

3. [optional - simple type restrictions only] Click the **Detail** tab and enter appropriate facets (constraints on the set of values of a simple type) for the restriction:

Icon	Facet
	Length - Exact number of characters or list items allowed. It must be equal to or greater than zero

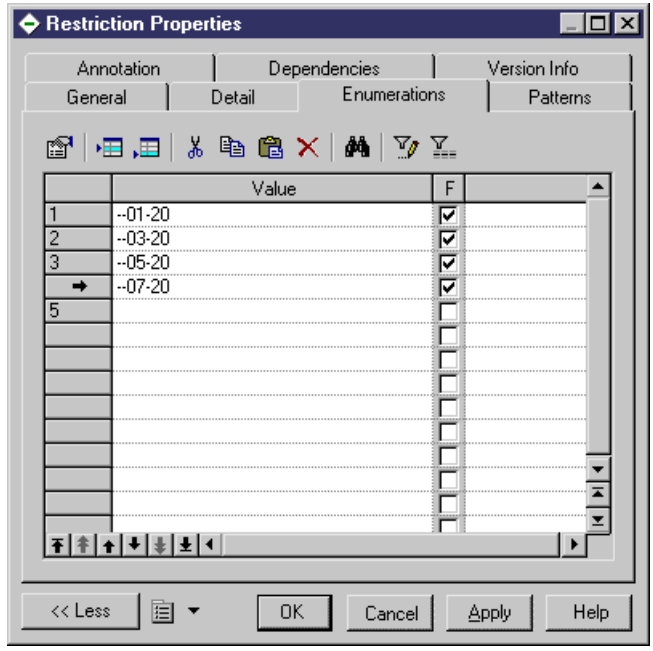
Icon	Facet
	Minimum length - Minimum number of characters or list items allowed. It must be equal to or greater than zero
	Maximum length - Maximum number of characters or list items allowed. It must be equal to or greater than zero
	Minimum exclusive - Lower bound for numeric values. All values are greater than this value
	Maximum exclusive - Upper bound for numeric values. All values are lower than this value
	Minimum inclusive - Minimum value allowed for data type
	Maximum inclusive - Maximum value allowed for data type
	Total digits - Exact number of decimal digits allowed. It must be greater than zero
	Fraction digits - Maximum number of decimal digits in the fractional part
	Whitespace - Way of handling white spaces. You can choose from the following: • *Preserve* - white spaces are unchanged. • *Replace* - Tabs, line feeds and carriage returns are replaced with spaces. • *Collapse* - Contiguous sequences of spaces are collapsed to a single space. Leading and trailing spaces are removed.

You can optionally click the **Properties** tool to the right of each field to open the property sheet of the facet and enter the following properties:

Property	Description
ID	ID of the facet. Its value must be of type ID and unique within the model containing the facet
Value	Value(s) of the facet
Fixed	To prevent a modification of the facet value(s), select the Fixed property

4. [optional - simple type restrictions only] Click the **Enumerations** tab and enter a set of acceptable values. Select the **F**[ixed] check box to prevent the modification of a value.

   For example: the meetings simple type, based on the xs:gMonthDay data type, is restricted to the following dates: 01/20, 03/20, 05/20 and 07/20.

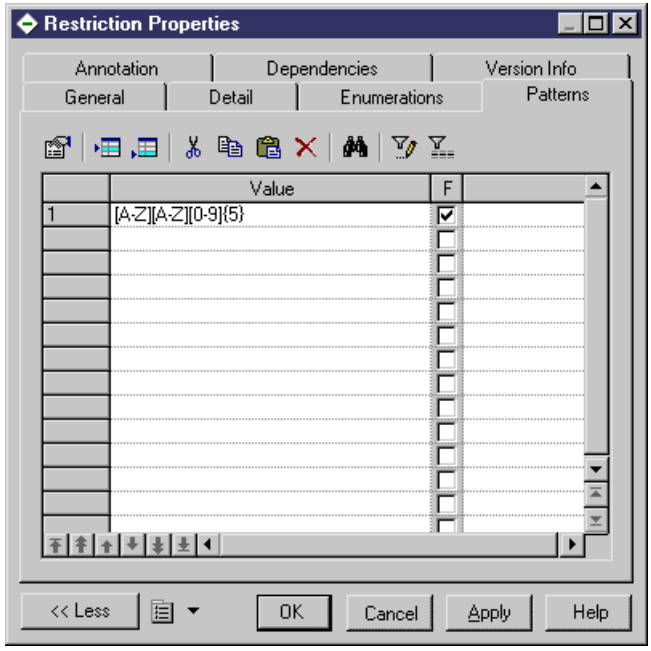


Generated schema:

```
<xs:simpleType name="MEETINGS">
  <xs:restriction base="xs:gMonthDay">
     <xs:enumeration value="--01-20"/>
     <xs:enumeration value="--03-20"/>
     <xs:enumeration value="--05-20"/>
     <xs:enumeration value="--07-20"/>
     <xs:enumeration/>
  </xs:restriction>
</xs:simpleType>
```

5. [optional - simple type restrictions only] Click the **Patterns** tab and enter one or more sequences of acceptable values. Select the **F**[ixed] check box to prevent the modification of a value.

   For example: the zipCode simple type, based on the xs:string data type, is restricted to the following pattern: two uppercase letters, from A to Z, followed by a five-digit number, each digit ranging from 0 to 9.



Generated schema:

```
<xs:simpleType name="ZIPCODE">
  <xs:restriction base="xs:string">
     <xs:pattern value="[A-Z][A-Z][0-9]{5}"/>
  </xs:restriction>
</xs:simpleType>
```

6. Click **OK** to close the restriction property sheet and return to the element, simple type, or complex type.

Deriving by List

You can derive an element or simple type by list to define it as a list of values of a specified data type.

1. Open the property sheet of an element or simple type, and select List in the Derivation list.

   For elements, the Embedded type field is set to Simple.

2. Click the Properties tool to the right of the Derivation box to open the list property sheet and complete the following properties:

Property	Description
ID	ID of the simple type list. Its value must be of type ID and unique within the model.

Property	Description
Type	Data type for the list of values
Embedded Type	If selected, the type disappears and a simple type is created in the schema within the current simple type or element. Click Apply, and then the Properties tool beside the Embedded type box, to define a derivation and a type for the embedded simple type.

3. Click OK to close the list property sheet and return to the element or simple type.

Deriving by Union

You can derive an element or simple type by union to define it as a collection of built-in and simple data types.

1. Open the property sheet of an element or simple type, and select Union in the Derivation list.

   For elements, the Embedded type field is set to Simple.

2. Click the Properties tool to the right of the Derivation box to open the union property sheet and complete the following properties:

Property	Description
ID	ID of the simple type union. Its value must be of type ID and unique within the model containing the simple type union.
Member Types	White space separated list of built-in data types. Values must be qualified names.

3. [optional] Click the Union Types tab and add appropriate simple types to the union.
4. Click OK to close the union property sheet and return to the element or simple type.

Annotations (XSM)

Annotations are only available in models targeted with XSD.

You define annotations when you want to add information about an XML model. There are three kinds of annotations:

- Annotation - provides extra information on XML models or schemas. You can define multiple annotations at this level, and each can contain multiple documentation and/or application information tags.
- Documentation – is contained within an annotation and contains an URI reference or any well-formed XML content that gives extra information about XML objects or documents
- Application Information - is contained within an annotation and contains an URI reference or any well-formed XML content that is used by applications for processing instructions

Generated schema of a global annotation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:annotation id="ANNOT1">
      <xs:documentation source="attributes.dtd" xml:lang="en-US">
        <?xml version="1.0" encoding="UTF-8" ?>
        <!ELEMENT COMPANY (EMPLOYEES,PRODUCTS,CLIENTS)>
        <!--  -->
        <!ATTLIST COMPANY
           FOUNDATION                    CDATA
           STATUS                        CDATA>
        <!ELEMENT EMPLOYEES EMPTY>
        <!--  -->
        <!ATTLIST EMPLOYEES
           NUMBER                        CDATA>
        <!ELEMENT PRODUCTS EMPTY>
        <!--  -->
        <!ATTLIST PRODUCTS
           REFERENCES                    CDATA
           QUANTITIES                    CDATA>
        <!ELEMENT CLIENTS EMPTY>
        <!--  -->
        <!ATTLIST CLIENTS
           FIDELITY                      CDATA
           SOLVENCY                      CDATA>
        </xs:documentation>
      <xs:appinfo source="http://www.parsersandco.com"></xs:appinfo>
   </xs:annotation>
</xs:schema>
```

This global annotation is composed of a documentation, with a well-formed XML content (extract of a DTD file), and an application information.

Creating an Annotation

You can create a annotation at the global level in any of the following ways:

- Open the Items or External Schemas tab in the property sheet of the model, and click the Add Annotation tool.
- Right-click the model or package in the Browser, and select **New > Annotation**.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Annotation Properties

You can modify an object's properties from its property sheet. To open an annotation property sheet, double-click its Browser entry in the Annotations folder. The following sections detail the property sheet tabs that contain the properties most commonly entered for annotations.

The General tab contains the following properties:

Property	Description
ID	Must be of type ID and unique within the model containing the annotation.

Items Tab

This tab lists the documentation and application information tags contained within the annotation. You can create a documentation or application information using the following tools:

Tool	Description
	Add Documentation - Adds a comment or a document reference to be read by users
	Add Application Information - Adds an information to be used by applications for processing instructions

Note: These tools are also available on the Annotations tab of an element or other object property sheet - to add content to an annotation at the global, schema level

Documentation and Application Information Properties

The General tab contains the following properties:

Property	Description
Source	Source of the content. It must be a URI reference
Language	[documentation only] Language used in the documentation. For example: en, en-GB, en-US, de, fr

The Content tab allows you to write or paste any well-formed XML content.

Notations (XSM)

Notations allow you to define and process non-XML objects within an XML model.

The following example shows the generated schema for a notation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <!--
   Integrating GIF files in your XML model
   -->
   <xs:notation name="PICTURES" public="pictures/gif"
 system="user/local/pictureViewer"/>
</xs:schema>
```

Notations are not available on models targeted with XDR.

Creating a Notation

You can create a notation in any of the following ways:

- Select **Model > Notations** to access the List of Notations, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select **New > Notation**.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Notation Properties

You can modify an object's properties from its property sheet. To open a notation property sheet, double-click its diagram symbol or its Browser entry in the Notations folder.

The General tab contains the following properties:

Property	Description
Name	Name of the notation, which must be a no-colon-name.
Code	Code of the notation, which must be a no-colon-name.
Comment	Descriptive label of the notation.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.
ID	ID of the notation. Its value must be of type ID and unique within the model containing the notation.
Public	URI reference identifying the non-XML object. For example: pictures/gif.
System	URI reference identifying the application that will process the non-XML object. For example: user/local/pictureViewer.

Entities (XSM)

Entities enable you to include predefined values, external XML or non-XML files in an XML model targeted with a DTD.

When an XML processor reads an entity reference in an XML document, it will replace this entity reference by its value defined in the DTD file of the XML document.

An entity reference is the entity name preceded by an ampersand and followed by a semicolon.

For example: `&furtherinfo;` will be replaced by `For further information, see`.

The W3C has predefined five entities for XML tags:

Entity name	Reference	Value
Less than	<	<
Greater than	>	>
Ampersand	&	&
Apostrophe	'	'
Quotation	"	"

In an XML model, you just need to type the name and the value of an entity.

Creating an Entity

You can create an entity in any of the following ways:

- Select **Model > Entities** to access the List of Entities, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select **New > Entity**.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Entity Properties

You can modify an object's properties from its property sheet. To open an entity property sheet, double-click its diagram symbol or its Browser entry in the Entities folder.

The General tab contains the following properties:

Property	Description
Name	Name of the entity, which must be a no-colon-name.
Code	Code of the entity, which must be a no-colon-name.
Comment	Descriptive label of the entity.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.
Value	Value of the entity. A string of characters in the case of a predefined value. A URI in the case of an XML or a non-XML file. For example: http://something.com/pictures/logo.gif.
Public	URI reference identifying the non-XML object. For example: pictures/gif.
System	URI reference identifying the application that will process the non-XML object. For example: user/local/pictureViewer.
Notation	Used to define and process non-XML objects within an XML model.

Property	Description
Parameter	If selected, the entity is parsed within the DTD, and not within the XML document as for a general entity. A parameter entity allows you to predefine a value within a DTD. This predefined value can then be easily changed within the DTD.

Instructions: Import, Include and Redefine (XSM)

Import, Include and Redefine allow you to enrich your XML model with external namespaces, schema files or schema components.

These instructions are only available in a model targeted with XSD.

Imports

An import identifies a namespace whose schema components are referenced by the current schema, allowing you to use components from any schema with different target namespace than the current schema.

In a schema, an import is declared with the <import> tag. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:import id="IMP1" namespace="xml.ordering"
 schemaLocation="ORDER.xsd"/>
</xs:schema>
```

Includes

An include allows you to include a specified schema file in the target namespace of the current schema, allowing you to use components from any schema with the same target namespace as the current schema or with no specified target namespace.

In a schema, an include is declared with the <include> tag. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:include id="INC1" schemaLocation="PROFORMA.xsd"/>
</xs:schema>
```

Redefines

A redefine allows you to redefine simple and complex types, groups and attribute groups from an external schema file in the current schema, allowing you to use components from any schema with the same target namespace as the current schema or with no specified target namespace.

In a schema, a redefine is declared with the <redefine> tag. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:redefine id="REDEF1" schemaLocation="CUSTOMERS.xsd">
      <xs:group name="PRIVILEGED_CUSTOMERS">
         <xs:all>
            <xs:element name="CUSTOMER_1"/>
            <xs:element name="CUSTOMER_2"/>
            <xs:element name="CUSTOMER_3"/>
         </xs:all>
      </xs:group>
      <xs:complexType name="PRIVILEGE">
      </xs:complexType>
   </xs:redefine>
</xs:schema>
```

Creating an Import, Include, or Redefine Instruction

You can create an import, include, or redefine instruction in any of the following ways:

- Select **Model > Import**, Include, or Redefine to access the relevant list, and click the Add a Row tool.
- Open the External Schemas tab in the property sheet of the model, and click the Add Import, Add Include, or Add Redefine tool.
- Right-click the model or package in the Browser, and select **New > Import**, Include, or Redefine.

For general information about creating objects, see the Objects chapter in the *Core Features Guide*.

Import, Include, and Redefine Properties

You can modify an import, include, or redefine instruction's properties from its property sheet. To open an instruction property sheet, double-click its diagram symbol or its Browser entry in the Imports, Includes, or Redefines folder. The following sections detail the property sheet tabs that contain the properties most commonly entered for instructions.

The General tab contains the following properties:

Property	Description
Schema location	URI reference for the location of a schema file with an external namespace. You can use the Browse tool beside the Properties tool to select a schema file among those opened in the current workspace. For example: ORDER.xsd.
ID	ID of the instruction. Its value must be of type ID and unique within the schema containing the instruction.
Namespace	[import only] URI reference for the namespace to import. For example: xml.ordering.
Comment	Descriptive label of the instruction.

Property	Description
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.

Items Tab

This tab is available for redefines only and lists the items to be redefined. The following tools are available:

Tool	Description
	Add Group - Adds a group of elements to be redefined.
	Add Attribute Group - Adds a group of attributes to be redefined.
	Add Simple Type - Adds a simple type to be redefined.
	Add Complex Type - Adds a complex type to be redefined.

Redefine Property Sheet Items Tab

The Items tab lists the items to be redefined:

Tool	Description
	Add Group - Adds a group of elements to be redefined.
	Add Attribute Group - Adds a group of attributes to be redefined.
	Add Simple Type - Adds a simple type to be redefined.
	Add Complex Type - Adds a complex type to be redefined.

Business Rules (XSM)

A business rule is a rule that your business follows. It is a written statement specifying what an information system must do or how it must be structured. It could be a government-imposed law, a customer requirement, or an internal guideline.

You can attach business rules to your model objects to guide and document the creation of your model. For example, the rule "an employee belongs to only one division" can help you graphically build the link between an employee and a division.

For more information, see "Business Rules" in the Objects chapter of the *Core Features Guide*.

CHAPTER 3 Generating and Reverse Engineering XML Schemas and Other Models

PowerDesigner supports the generation and reverse-engineering of XML Schema Definition files (.XSD), Document Type Definition files (.DTD) and XML-Data Reduced files (.XDR). You can also generate a physical data model (PDM) from an XSM or generate an XSM from a PDM

Generating XML Schema Files

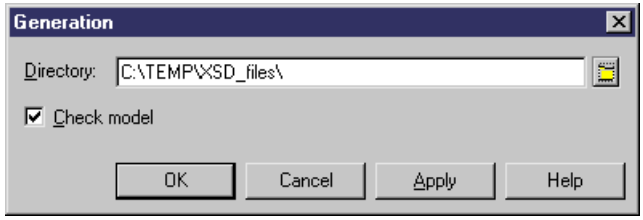
PowerDesigner provides a standard interface for generating all the supported XML schemas.

Target Schema	Generated file
XML Schema Definition 1.0	XSD
Document Type Definition 1.0	DTD **Note:** Parameter entities are references to predefined values within a DTD file (See Parameter property in entity property sheet). During DTD generation, some object properties containing inadvertently parameter values will be generated with parameter references. If you are not satisfied with this default use of parameter entities, you should clear the Parameter property before generation.
XML-Data Reduced 1.0	XDR

You can preview the file to be generated by selecting the Preview tab of your XML model property sheet (see *Previewing XML Code* on page 8).

Note: The PowerDesigner generation system is extremely customizable through the use of extensions (see *Extending your Modeling Environment* on page 11). For detailed information about customizing generation, including adding generation targets, options, and tasks, see *Chapter 1, Resource Files and the Public Metamodel* in *Customizing and Extending PowerDesigner*

1. Select **Language > Generate*schema* File** to open the Generation dialog:



2. Enter a directory in which to generate the files and specify whether you want to perform a model check. For more information about checking your model, see *Chapter 4, Checking an XSM* on page 83.

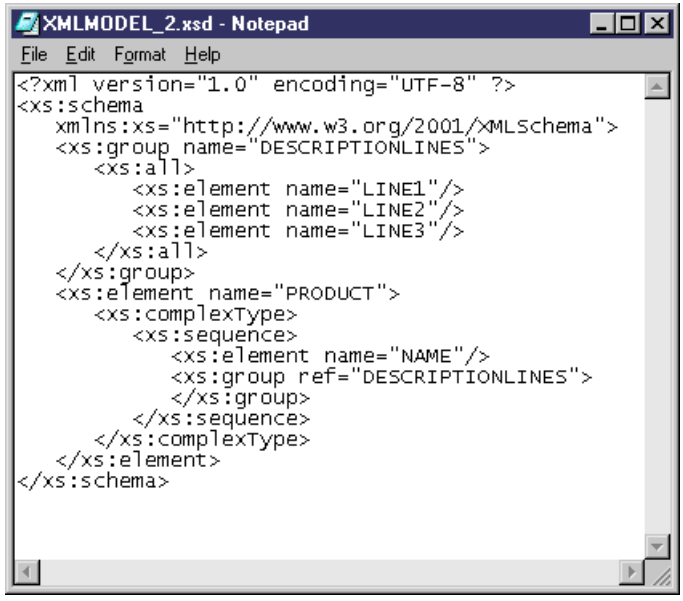
Note: When generating an XDR file, the Generation dialog contains an Options tab, where you can specify whether or not to generate comments (within a <description> tag). This option is enabled by default.

---

3. Click OK to begin generation.

   A Progress box is displayed. The Result list displays the files that you can edit. The result is also displayed in the Generation tab of the Output window, located in the bottom part of the main window.

4. Click Edit to edit the XSD, DTD or XDR file in your associated editor:



Reverse Engineering an XML Schema into an XSM

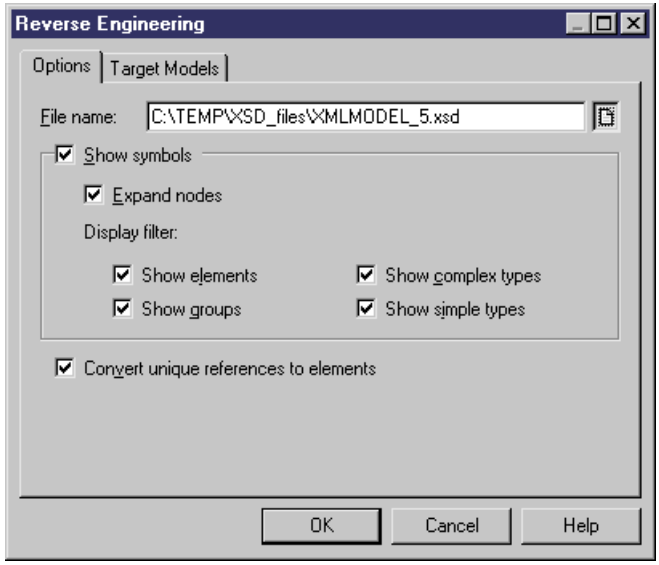
Reverse engineering is the process of extracting an XML structure from an XML schema file, and using it to build or update an XSM.

Note: PowerDesigner uses a parser software for XML reverse engineering, developed by the Apache Software Foundation (*http://www.apache.org*).

Reverse Engineering to a New XML Model

You can reverse engineer XML schema files to create a new XSM.

1. Select **File > Reverse Engineer > XML Definition** to open the New XML Model dialog box.
2. Select an XML language and specify whether you want to share the resource file or copy it to your model.
3. [optional] Click the **Select Extensions** tab, and select any extension files you want to attach to the new model.
4. Click **OK** to go to the Reverse Engineering dialog:



5. On the **Options** tab, specify the file you want to reverse engineer, and select any appropriate options:

Option	Description
Show symbols	Creates symbols for the reversed objects in the diagram. If you select to show symbols, you can also specify to expand all the nodes, and to display elements, groups, and complex and simple types.
Convert unique references to elements	Enables the display of shortcuts to XML structures in other models as expandable nodes, instead of simple shortcuts. Since global objects with a single reference in the model will be converted into child objects, you should not use this option if you want to keep the global scope of some objects. You can subsequently perform this conversion by selecting **Tools > Convert Unique References** in the XML model.

6. [optional] Click the **Target Models** tab and specify any existing PowerDesigner models which are referenced in the file being reverse engineered. These references will become shortcuts in the reversed model.
7. Click **OK** to begin reverse engineering.

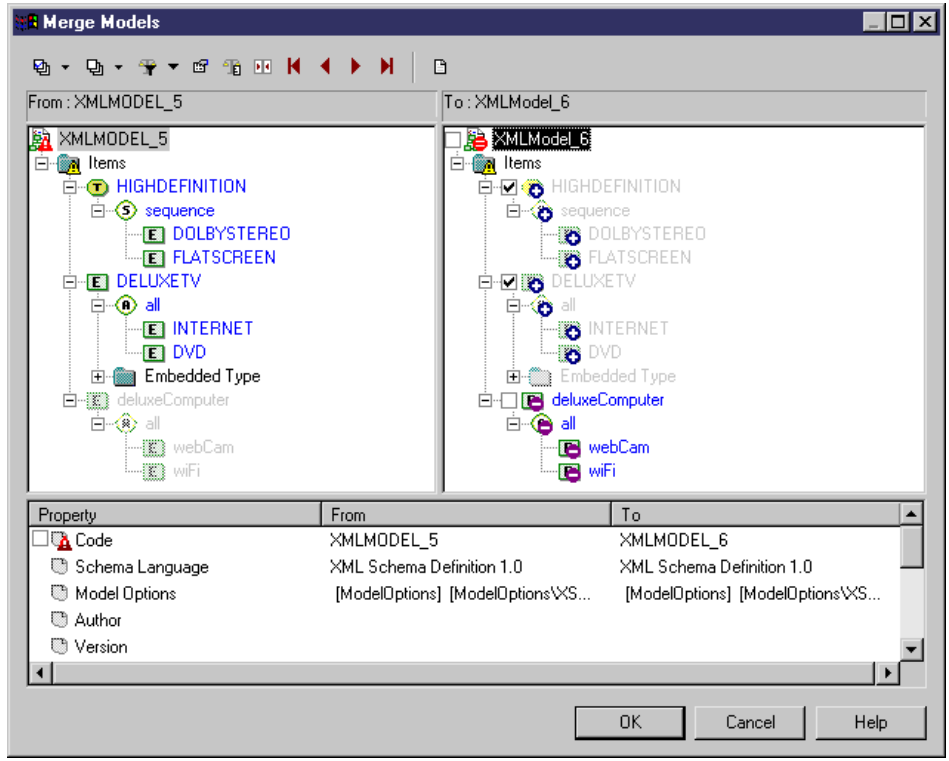
   The XML file is reversed into an XML model and displayed in the diagram window and the Browser. The result is also displayed in the Reverse page of the Output window.

Reverse Engineering to an Existing XML Model

You can reverse engineer XML schema files to add objects to an existing XSM.

1. Open the XML model you want to reverse into and then select **Language > Reverse Engineer *schema* File** to open the Reverse Engineering dialog box.
2. Select the file to reverse, and specify any appropriate options.
3. [optional] Click the Target Models tab and specify any existing PowerDesigner models which are referenced in the file being reverse engineered. These references will become shortcuts in the reversed model.
4. Click OK to begin reverse engineering.

   A message in the Output window confirms that the file has been reversed and the Merge Models window opens:



5. Review the objects that you will be importing, and the changes that they will make to the model (see *Chapter 7, Comparing and Merging Models* in the *Core Features Guide*).
6. Click OK to merge the selected changes into your model.
7. [optional] Select **Symbol > Auto-Layout** to organize the new symbols in your diagram.

Generating Other Models from an XSM

You can generate physical data models (PDMs) and other XSMs from an XSM.

1. Select **Tools**, and then one of the following commands to open the appropriate Model Generation Options window:
   - **Generate Physical Data Model... Ctrl+Shift+P**
   - **Generate XML Model... Ctrl+Shift+M**
2. On the **General** tab, select a radio button to generate a new or update an existing model, and complete the appropriate options.
3. [optional] Click the **Detail** tab and set any appropriate options. We recommend that you select the **Check model** checkbox to check the model for errors and warnings before generation.

4. [optional] Click the **Target Models** tab and specify the target models for any generated shortcuts.
5. [optional] Click the **Selection** tab and select or deselect objects to generate.
6. Click **OK** to begin generation.

Note: For detailed information about the model generation feature, see *Chapter 10, Generating Models and Model Objects* in the *Core Features Guide* .

The following table details how XSM objects are generated to PDM objects:

XSM	PDM
Elements	Tables or columns: • Root elements - are generated as tables. • Non-root elements with complex types - are generated as tables or columns, depending on the option chosen in the **Persistent** group-box on the **Detail** tab of the element property sheet. • Non-root elements with primitive or simple types - are generated as table columns. **Note:** Root elements with a primitive or simple type are not generated except where they are referenced by other elements or complex types. If you have a single root element and want to generate its immediate children as tables, select the **Skip single root element** option on the PDM Generation Options window **Detail** tab.
Simple types	Domains. The datatype of the domain depends on the derivation of the simple type: • simple types with a list derivation - `varchar`. • simple types with a restriction derivation - the datatype of the base type • simple types with a union derivation - the most permissive of the unioned types
Complex types	Merged with their parent element. If the complex type is the restriction or extension of a simple type it will be generated as a column called `Value` linked to the domain generated from the simple type.
Attributes	Columns with datatypes determined by resolving any derivation. Attributes and attribute groups defined at the model level are not generated except where they are referenced.

XSM	PDM
Business rules	Business rules
Key constraints	Keys
Unique constraints	Indexes
Keyref constraints	References (if the referenced constraint is a key)
IDs (DTD)	Keys

Note: References, substitutions, imports, and includes are always resolved, and attributes and attribute groups defined at the model level are generated only where they are used. Notations, redefines, anys, and (for DTDs) entities, are not generated to PDMs.

CHAPTER 4 Checking an XSM

The XML model is a very flexible tool, which allows you quickly to develop your model without constraints. You can check the validity of your XSM at any time.

A valid XSM conforms to the following kinds of rules:

- Each complex type should have at least one attribute
- Each group must contain elements, groups, group particles and/or Any

Note: We recommend that you check your XML model before generating an XML document or another model from it . If the check encounters errors, generation will be stopped. The Check model option is enabled by default in the Generation dialog box.

You can check your model in any of the following ways:

- Press F4, or
- Select **Tools** > **Check Model**, or
- Right-click the diagram background and select Check Model from the contextual menu

The Check Model Parameters window opens, allowing you to specify the kinds of checks to perform, and the objects to apply them to. For detailed information about this window and correcting problems reported, see "Checking a Model" in the Models chapter of the *Core Features Guide*.

The following sections document the XSM -specific checks available by default. For information about checks made on generic objects available in all model types, see "Checking a Model" in the Models chapter of the *Core Features Guide*.

Group Particle Checks

PowerDesigner provides default model checks to verify the validity of group articles.

Check	Description and Correction
Existence of particle	A group particle must contain elements, groups, group particles and/or Any. Manual correction: Add items to the group particle or delete it Automatic correction: None

Check	Description and Correction
Invalid cardinality	You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence. This check is only available in a model targeted with XDR. Manual correction: Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties Automatic correction: None

Model Checks

PowerDesigner provides default model checks to verify the validity of models built on a schema.

Check	Description and Correction
Identifier uniqueness	Two or more objects cannot have the same identifier (ID). Manual correction: Give a unique identifier to each object Automatic correction: None
Undefined identifier	You must define an identifier (ID) for each object in the model. Manual correction: Define an identifier for each object Automatic correction: None
Shortcut code uniqueness	Two shortcuts with the same code cannot be in the same namespace. Manual correction: Change the code of one of the shortcuts Automatic correction: None
Undefined target namespace	You should define a target namespace to your model. Manual correction: Type a URI for the Target Namespace property in the Detail tab of the model property sheet Automatic correction: None
Missing namespaces	There should be at least one namespace defined for the model. Manual correction: Type a URI and a prefix in the Namespaces tab of the model property sheet Automatic correction: Adds the target namespace URI and a prefix "ns" followed by a number (e.g. "ns1")

Data Source Checks

PowerDesigner provides default model checks to verify the validity of data sources.

Check	Description and Correction
Data source name and code uniqueness	Data sources names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Existence of model	A data source must have at least one model in its definition. Manual correction: Add a model from the Models tab of the data source property sheet Automatic correction: Deletes data source without a model
Data source containing models with different Object Language or DBMS types	The models in a data source represent a single set of information. This is why the models in the data source should share the same DBMS or object language. Manual correction: Delete models with different DBMS or object language, or modify the DBMS or object language of models in the data source Automatic correction: None

Entity Checks

PowerDesigner provides default model checks to verify the validity of entities.

Check	Description and Correction
Entity name and code uniqueness	Entity names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined entity	You must define an entity. In the entity property sheet, you must either type a value (string of characters or URI) in the Value box, or a URI in the Public or System boxes. Manual correction: Type a value in the Value box or a URI in the Public or System boxes Automatic correction: None

Include Checks

PowerDesigner provides default model checks to verify the validity of includes.

Check	Description and Correction
Undefined schema location	You must define a schema location for an include. Manual correction: Define a URI or select a schema file for the schema location. For example: proforma.xsd Automatic correction: None

Simple Type Checks

PowerDesigner provides default model checks to verify the validity of simple types.

Check	Description and Correction
Simple type name and code uniqueness	Simple type names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code

Complex Type Checks

PowerDesigner provides default model checks to verify the validity of complex types.

Check	Description and Correction
Complex type name and code uniqueness	Complex type names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Existence of attribute	A complex type should have at least one attribute. Manual correction: Define an attribute for the complex type Automatic correction: None

Check	Description and Correction
Existence of particle	A complex type must contain elements, groups, group particles and/or Any. Manual correction: Add items to the complex type or delete complex type Automatic correction: None

Element Checks

PowerDesigner provides default model checks to verify the validity of elements.

Check	Description and Correction
Element name and code uniqueness	Element names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined type	An element without a reference should have a defined data type. Manual correction: In the element property sheet, define a data type with the Type list or the Browse tool Automatic correction: None
Undefined reference	An element without a defined data type must have a reference. Manual correction: In the element property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Existence of attribute	An element without a reference, a data type or a substitution group should have at least one attribute. Manual correction: Define an attribute for the element Automatic correction: None
Existence of particle	An element with an embedded complex type must contain child elements, groups, group particles and/or Any. Manual correction: Add items to complex element or delete complex element Automatic correction: None

Check	Description and Correction
Invalid cardinality	[only available for model targeted with XDR] You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence. Manual correction: Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties Automatic correction: None

Group Checks

PowerDesigner provides default model checks to verify the validity of groups.

Check	Description and Correction
Group name and code uniqueness	Group names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined reference	A group without a name or a code must have a reference. Manual correction: In the group property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Existence of group particle	A group must contain elements, groups, group particles and/or Any. Manual correction: Add items to group or delete group Automatic correction: None
Invalid cardinality	[only available for model targeted with XDR] You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence. Manual correction: Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties Automatic correction: None

Attribute Checks

PowerDesigner provides default model checks to verify the validity of attributes.

Check	Description and Correction
Attribute name and code uniqueness	Attribute names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined reference	An attribute without a name or a code must have a reference. Manual correction: In the attribute property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Undefined type	You must define a data type for an attribute. Manual correction: In the attribute property sheet, define a data type with the Type list or the Browse tool Automatic correction: None

Notation Checks

PowerDesigner provides default model checks to verify the validity of notations.

Check	Description and Correction
Notation name and code uniqueness	Notations names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined notation	A notation must have at least one URI defined for Public or System properties. Manual correction: In the notation property sheet, define a URI in the Public or System boxes Automatic correction: None

Attribute Group Checks

PowerDesigner provides default model checks to verify the validity of attribute groups.

Check	Description and Correction
Attribute group name and code uniqueness	Attribute group names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined reference	An attribute group without a name or a code must have a reference. Manual correction: In the attribute group property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Existence of attributes	An attribute group must contain at least one attribute. Manual correction: Add attributes to attribute group or delete attribute group Automatic correction: Deletes unassigned attribute group

Import Checks

PowerDesigner provides default model checks to verify the validity of imports.

Check	Description and Correction
Undefined schema location and namespace	An import must have at least a schema location or a namespace defined. Manual correction: In the import property sheet, define a URI for the schema location and/or the namespace. Automatic correction: None

Redefine Checks

PowerDesigner provides default model checks to verify the validity of redefines.

Check	Description and Correction
Undefined schema location	You must define a schema location for a redefine. Manual correction: In the redefine property sheet, define a URI or select a schema file for the schema location. For example: customers.xsd Automatic correction: None
Existence of component	A redefine must contain at least one of the following items: simple type, complex type, group or attribute group. Manual correction: Add items to the redefine Automatic correction: None

Key Checks

PowerDesigner provides default model checks to verify the validity of keys.

Check	Description and Correction
Key name and code uniqueness	Element names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Existence of fields	A key must contain at least one field. Manual correction: Add at least one field to the key or delete the key. For example: @numEmployee Automatic correction: Deletes unassigned key For more information on fields, see *Specifying Constraint Fields* on page 45.

Check	Description and Correction
Undefined selector	You must define an XPath expression for a key selector attribute. Manual correction: In the key property sheet, define an XPath expression for the selector attribute. For example: s:company/s:employee Automatic correction: None For more information on XPath expressions, see *Specifying a Constraint Selector* on page 45.

KeyRef Checks

PowerDesigner provides default model checks to verify the validity of KeyRefs.

Check	Description and Correction
KeyRef name and code uniqueness	KeyRef names and codes must be unique in the model. Manual correction: Modify the duplicate name/code. Automatic correction: Appends a number to the duplicate name/code.
Undefined reference	A keyRef must contain a reference to a key or a unique constraint. Manual correction: In the keyRef property sheet, define a reference to a key or a unique constraint with the Reference list. Automatic correction: None
Existence of fields	A keyRef must contain at least one field. Manual correction: Add at least one field to the keyRef or delete the keyRef. For example: @numEmployee. Automatic correction: Deletes unassigned keyRef. For more information on fields, see *Specifying Constraint Fields* on page 45.
Undefined selector	You must define an XPath expression for a keyRef selector attribute. Manual correction: In the keyRef property sheet, define an XPath expression for the selector attribute. For example: s:company/s:employee. Automatic correction: None For more information on XPath expressions, see *Specifying a Constraint Selector* on page 45.

Unique Checks

PowerDesigner provides default model checks to verify the validity of uniques.

Check	Description and Correction
Unique name and code uniqueness	Unique constraint names and codes must be unique in the model. Manual correction: Modify the duplicate name/code. Automatic correction: Appends a number to the duplicate name/code.
Existence of fields	A unique constraint must contain at least one field. Manual correction: Add at least one field to the unique constraint or delete the unique constraint. For example: @numEmployee. Automatic correction: Deletes unassigned unique constraint. For more information on fields, see *Specifying Constraint Fields* on page 45.
Undefined Selector	You must define an XPath expression for a unique constraint selector attribute. Manual correction: In the unique constraint property sheet, define an XPath expression for the unique constraint selector attribute. For example: s:company/s:employee. Automatic correction: None For more information on XPath expressions, see *Specifying a Constraint Selector* on page 45.

Extension Checks

PowerDesigner provides default model checks to verify the validity of extensions.

Check	Description and Correction
Undefined base type	You must define a base type when you derive a complex type by extension. Manual correction: In the complex type property sheet, click the Properties tool beside the Derivation box to display the Extension property sheet and select a base type with the Base type list or the Browse tool Automatic correction: None

Restriction Checks

PowerDesigner provides default model checks to verify the validity of restrictions.

Check	Description and Correction
Undefined base type	You must define a base type when you derive a simple or a complex type by restriction. Manual correction: In the simple or complex type property sheet, click the Properties tool beside the Derivation box to display the Extension property sheet and select a base type with the Base type list or the Browse tool Automatic correction: None
Existence of facet	A simple type restriction must have at least one facet defined. Facets are defined in the Detail, Enumerations and Patterns tabs of a simple type restriction property sheet. Manual correction: Define one or more facets in the simple type restriction property sheet Automatic correction: None

Simple Type List Checks

PowerDesigner provides default model checks to verify the validity of simple types.

Check	Description and Correction
Undefined base type	You must define a base type when you derive a simple type by list. Manual correction: In the simple type property sheet, click the Properties tool beside the Derivation box to display the simple type list property sheet and select a data type with the Type list or the Browse tool Automatic correction: None

Simple Type Union Checks

PowerDesigner provides default model checks to verify the validity of simple type unions.

Check	Description and Correction
Undefined base type	You must define at least two data types when you derive a simple type by union. Manual correction: In the simple type property sheet, click the Properties tool beside the Derivation box to display the simple type union property sheet and type a white space separated list of at least two data types (qualified names) in the Member types box Automatic correction: None

Annotation Checks

PowerDesigner provides default model checks to verify the validity of annotations.

Check	Description and Correction
Existence of items	An annotation must contain at least one URI for a Documentation or an Application Information. Manual correction: Define a URI for a Documentation or an Application Information Automatic correction: None

CHAPTER 5 Working with XML and Databases

Many relational databases now support XML so that you can store or retrieve data through XML files. You can use an XML model to generate an *annotated schema* that will allow you to store or retrieve data in such a database.

The following databases are available :

Database	Mapped XML model	Targeted XML language	Required XEM file
Microsoft SQL Server 2000 and higher	Yes	XSD or XDR	Microsoft SQL Server
Oracle 9i2 and higher	No	XSD	Oracle 9i2
IBM DB2 v8.1 and higher	Yes	DTD	IBM DB2 DAD

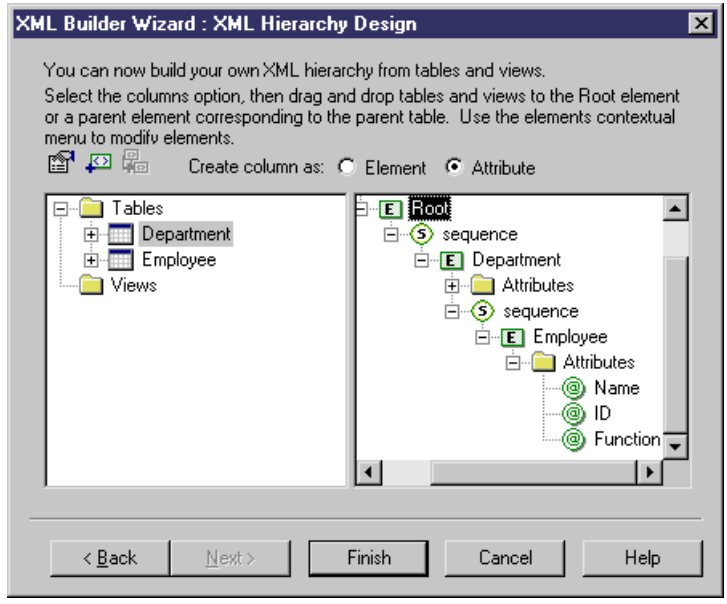
By attaching the SQL/XML extensions to an XML model mapped to a PDM, you can also generate *SQL/XML queries* to retrieve data in an XML format, from relational databases supporting SQL/XML.

Note: You can also generate PDM tables from an XML schema. For more information, see *Generating Other Models from an XSM* on page 79.

Mapping Database Objects to an XML Schema Via the XML Builder Wizard

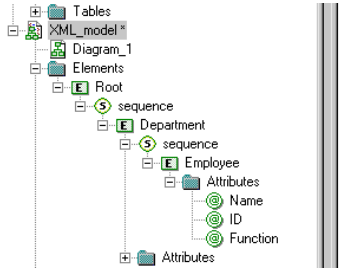
You can create an XSM to generate an annotated schema from a PDM via the XML Builder.

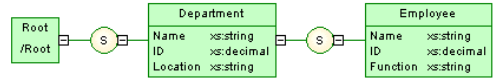
1. Open a PDM targeted with the appropriate DBMS, and select **Tools > XML Builder Wizard** to open the XML Builder Wizard.
2. Specify whether to create a new XML model or work with an existing model, and then click **Next** to go to the Tables and Views Selection page.
3. Select the tables and views from which you want to generate the schema, and then click **Next** to go to the XML Hierarchy Design page.
4. Build your hierarchy by dragging and dropping tables and/or columns from the left pane to the right pane or by using the tools above the panes:



For detailed information about using the wizard, see "Generating an XSM from a PDM via the XML Builder Wizard" in the Working with Data Models chapter of the *Data Modeling* guide.

5. Click Finish to generate the XML model:





In the case of an existing XML model, the generated elements appear alongside the existing elements.

Note: The `SQL/XML` extension file is automatically attached to the generated XML model. You can optionally attach the `XML Document` extension file to generate a simplified XML file that will help you understand the annotated schema (see *Extending your Modeling Environment* on page 11).

Generating an SQL/XML Query File

SQL/XML is an XML extension of the Structured Query Language, which allows you to retrieve relational data using extended SQL syntax, and produce an XML result. You can generate SQL/XML queries for *global elements* in your XSM, whatever the targeted XML language (XSD, DTD or XDR).

SQL/XML has five main elements:

- *XMLELEMENT* - to edit an element with a name, a list of attributes (optional) and a list of values (optional)
- *XMLATTRIBUTES* - to edit a list of attributes with names and values
- *XMLAGG* - to edit in multiple rows a concatenation of elements, from a single XML value corresponding to a single column
- *XMLCONCAT* - to edit in the same row a concatenation of elements, from several XML values corresponding to several columns
- *XMLFOREST* - to edit in the same row a concatenation of elements, from several SQL values corresponding to several columns. The name and value of a column become the name and value of an element

You can generate an SQL/XML query file from an XSM if you have attached the SQL/XML extension file. These extensions are automatically linked to your XSM if you have created it from a PDM via the XML Builder Wizard (see *Mapping Database Objects to an XML Schema Via the XML Builder Wizard* on page 97). If need be, you can still modify the mapping through the Mapping tab of elements and complex types property sheets.

To manually enable the SQL/XML extensions in your model, select **Model > Extensions**, click the **Import** tool, select the `SQL/XML` file (on the **General Purpose** tab), and click **OK** to attach it.

Warning! The following procedure assumes you have an XML model open in the workspace and mapped to a PDM. Generated SQL/XML queries cannot be parameterized.

1. Select **Tools > Generate SQL/XML Queries** to open the Generation dialog box.
2. Specify the directory in which to generate the file.
3. Click the Selection tab and specify which of the global elements you want to generate queries from. A separate file will be generated for each global element selected.
4. Click OK to begin the generation.

   The Result dialog box is displayed with the path of the query file selected.
5. Click Edit to open the generated query file in your associated editor:

```
select '<?xml version="1.0" encoding="UTF-8" ?>' ||
XMLELEMENT( NAME "Root",
(select XMLAGG ( XMLELEMENT( NAME "DEPARTMENT", XMLATTRIBUTES (DEPARTMENT.DEPNUM
 (select XMLAGG ( XMLELEMENT( NAME "EMPLOYEE", XMLATTRIBUTES (EMPLOYEE.DEPNUM AS
  from EMPLOYEE
  where DEPARTMENT.DEPNUM = EMPLOYEE.DEPNUM)) )
 from DEPARTMENT))
```

Generating an Annotated Schema for Microsoft SQL Server

Microsoft SQL Server is an XML-enabled database server, which supports annotations that can be used on XSD or XDR files, to map XML data to relational data.

An *annotated schema* is an XML file that allows you to store or retrieve data in an XML format, from relational databases supporting XML. An XML model allows you to generate an annotated schema (XSD or XDR) for SQL Server 2000.

1. Map an XSM to a PDM. You can do this manually or by generating an XSM from a PDM (or a PDM from an XSM) but we recommend that you use the XML Builder Wizard (see *Mapping Database Objects to an XML Schema via the XML Builder Wizard* on page 97)
2. [if you do not use the wizard] Attach the Microsoft SQL Server extension file. To enable these extensions in your model, select **Model > Extensions**, click the **Import** tool, select the `Microsoft SQL Server` file (on the **XML in Database** tab), and click **OK** to attach it.
3. [optional] Reinforce the mappings of elements and attributes to tables and columns with extended attributes:

Note: If the element and attribute names match the table and column names, you do not need to define extended attributes for XML objects.

Annotation	Description
encode	When an XML element or attribute is mapped to a SQL Server BLOB column, allows requesting a reference (URI) to be returned and used later to return BLOB data. Available for: Element, Attribute
field	Maps an XML item to a database column. Available for: Element, Attribute
hide	Hides the element or attribute specified in the schema in the resulting XML document. Available for: Element, Attribute

Annotation	Description
is-constant	Creates an XML element that does not map to any table. The element is displayed in the query output. Available for: Element
key-fields	Allows specification of columns that uniquely identify the rows in a table. Available for: Element
limit-field	Allows limiting the values that are returned on the basis of a limiting value. Available for: Element, Attribute
limit-value	Allows limiting the values that are returned on the basis of a limiting value. Available for: Element, Attribute
mapped	Allows schema items to be excluded from the result. Available for: Element, Attribute
max-depth	Allows you to specify depth in recursive relationships that are specified in the schema. Available for: Element
overflow-field	Identifies the database column that contains the overflow data. Available for: Element
relation	Maps an XML item to a database table. Available for: Element
relationship-child	Specifies an element as the *child table* in a reference (To define only in the child element property sheet). Available for: Element
relationship-child-key	Specifies an attribute as the *foreign key* of a child table in a reference (To define only in the child element property sheet). Available for: Element
relationship-parent	Specifies an element as the *parent table* in a reference (To define only in the child element property sheet). Available for: Element
relationship-parent-key	Specifies an attribute as the *primary key* of a parent table in a reference (To define only in the child element property sheet). Available for: Element

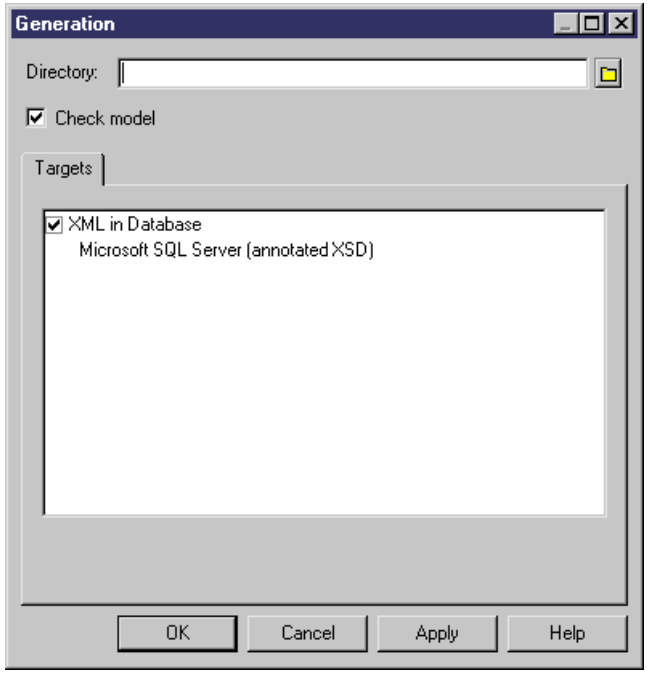
Annotation	Description
use-cdata	Allows specifying CDATA sections to be used for certain elements in the XML document. Available for: Element
prefix	Creates valid XML ID, IDREF, and IDREFS. Prepends the values of ID, IDREF, and IDREFS with a string. Available for: Attribute

4. [optional] Click the **Preview** tab of the model property sheet, to preview the annotated schema.
5. Generate the annotated schema (see *Generating the SQL Server Annotated Schema File* on page 102).

Generating the SQL Server Annotated Schema File

You generate the annotated schema file by selecting it as an additional target for standard schema generation.

1. Select **Language > Generate *schema*File** to open the Generation dialog box.
2. Specify the directory in which to generate the file and select the XML in Database target on the Targets tab.



3. Click OK to begin the generation.

   The Result dialog box is displayed with the path of the annotated schema file selected.

4. Click Edit to open the generated annotated schema in your associated editor:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
   <xs:element name="Root" sql:is-constant="1">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="DEPARTMENT" sql:relation="DEPARTMENT">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="EMPLOYEE" sql:relation="EMPLOYEE">
                        <xs:annotation>
                           <xs:appinfo>
                              <sql:relationship parent="DEPARTMENT" parent-key="DEPNUM" child-key="DEPNUM" child="EMPLOYEE"/>
                           </xs:appinfo>
                        </xs:annotation>
                        <xs:complexType>
                           <xs:attribute name="DEPNUM" type="xs:int" sql:field="DEPNUM">
                           </xs:attribute>
                           <xs:attribute name="EMPID" type="xs:int" sql:field="EMPID">
                           </xs:attribute>
                           <xs:attribute name="FIRSTNAME" type="xs:string" sql:field="FIRSTNAME">
                           </xs:attribute>
                           <xs:attribute name="LASTNAME" type="xs:string" sql:field="LASTNAME">
                           </xs:attribute>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
                  <xs:attribute name="DEPNUM" type="xs:int" sql:field="DEPNUM">
                  </xs:attribute>
                  <xs:attribute name="DEPNAME" type="xs:string" sql:field="DEPNAME">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

   Note the SQL namespace (with the sql prefix) and the SQL annotations for tables (sql:relation), columns (sql:field) and reference (sql:relationship).

Generating an Annotated Schema for Oracle 9i2

Oracle 9i2 is a database server with a native XML storage and retrieval technology called *Oracle XML DB*. There is no mapping between XML data and relational data. Tables, columns and abstract data types (ADT) are created from *annotated schemas* (XSDs). Annotated schemas are XML-coded files, targeted with an XML language and tagged with specific DBMS annotations, that allow you to store or retrieve data in an XML format, from relational databases supporting XML.

An XML model allows you to generate an annotated schema (XSD) for Oracle 9i2. Oracle 9i2 uses by default the name of the XML elements present in the annotated schema to generate SQL objects. You can override the creation of SQL objects by defining *extended attributes* for elements, complex types and the XML model.

To enable these extensions in your model, select **Model > Extensions**, click the **Import** tool, select the `Oracle XML DB` (on the **XML in Database** tab), and click **OK** to attach it.

Oracle Extended Attributes for Elements and Attributes

You can set extended attributes on various XSM objects to define mappings with an Oracle database.

Element Extended Attributes

The following annotations can be specified on the Extended Attributes tab of the property sheets of elements:

Annotation	Description
beanClassname	Can be used within element declarations. If the element is based on a global complexType, this name must be identical to the beanClassname value within the complexType declaration. If a name is specified by the user, the bean generation will generate a bean class with this name, instead of generating a name from the element name
columnProps	Specifies the column storage clause that is inserted into the default CREATE TABLE statement. It is useful mainly for elements that are mapped to tables, namely top-level element declarations and out-of-line element declarations
defaultTable	Specifies the name of the table into which XML instances of this schema should be stored. This is most useful in cases when the XML is being inserted from APIs where table name is not specified (for example, FTP and HTTP)
javaClassname	Used to specify the name of a Java class that is derived from the corresponding bean class, to ensure that an object of this class is instantiated during bean access. If a JavaClassname is not specified, Oracle XML DB will instantiate an object of the bean class directly
maintainDOM	If true, instances of this element are stored so that they retain DOM fidelity on output. This implies that all comments, processing instructions, namespace declarations, and so on, are retained in addition to the ordering of elements. If false, the output need not be guaranteed to have the same DOM behavior as the input
maintainOrder	If true, the collection is mapped to a VARRAY. If false, the collection is mapped to a NESTED TABLE
SQLCollSchema	Name of the database user owning the type specified by SQLCollType
SQLCollType	Specifies the name of the SQL collection type corresponding to this XML element that has maxOccurs > 1
SQLInline	If true this element is stored inline as an embedded attribute (or a collection if maxOccurs > 1). If false, a REF (or collection of REFs if maxOccurs > 1) is stored. This attribute will be forced to false in certain situations (like cyclic references) where SQL will not support inlining

Annotation	Description
SQLName	Specifies the name of the attribute within the SQL object that maps to this XML element
SQLSchema	Name of the database user owning the type specified by SQLType
SQLType	Specifies the name of the SQL type corresponding to this XML element declaration
tableProps	Specifies the TABLE storage clause that is appended to the default CREATE TABLE statement. This is meaningful mainly for global and out-of-line elements

Complex Type Extended Attributes

The following annotations can be specified on the Extended Attributes tab of the property sheets of complex types:

Annotation	Description
beanClassname	Can be used within element declarations. If the element is based on a global complexType, this name must be identical to the beanClassname value within the complexType declaration. If a name is specified by the user, the bean generation will generate a bean class with this name, instead of generating a name from the element name
SQLSchema	Name of the database user owning the type specified by SQLType
SQLType	Specifies the name of the SQL type corresponding to this XML element declaration

Model Extended Attributes

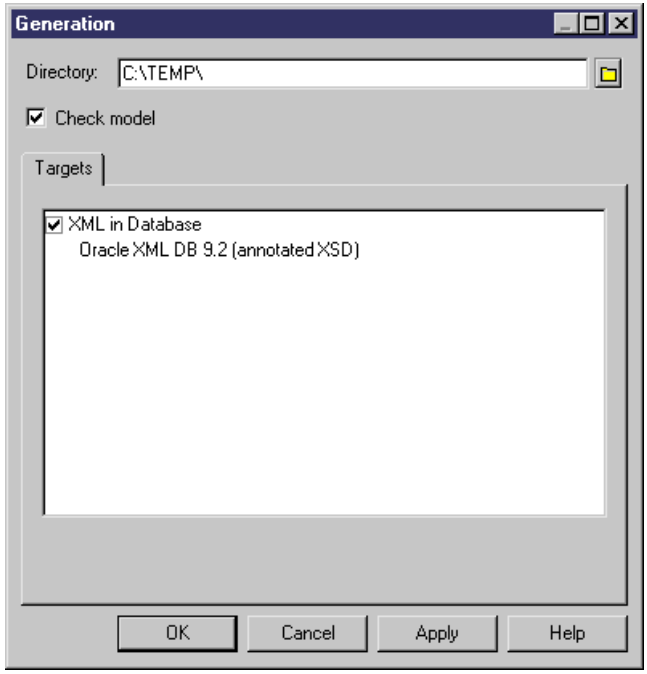
The following annotations can be specified on the Extended Attributes tab of the property sheet of the model:

Annotation	Description
mapUnboundedStringToLob	If true, unbounded strings are mapped to CLOB by default. Similarly, unbounded binary data get mapped to BLOB, by default. If false, unbounded strings are mapped to VARCHAR2(4000), and unbounded binary components are mapped to RAW(2000)
storeVarrayAsTable	If true, the VARRAY is stored as a table (OCT). If false, the VARRAY is stored in a LOB

Generating the Oracle Annotated Schema File

You generate the annotated schema file by selecting it as an additional target for standard schema generation.

1. Select **Language > Générate** *schema* File to open the Generation dialog box.
2. Specify the directory in which to generate the file and select the XML in Database target on the Targets tab.



3. Click OK to begin the generation.

   The Result dialog box is displayed with the path of the annotated schema file selected.

4. Click Edit to open the generated annotated schema in your associated editor:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:sql="http://xmlns.oracle.com/xdb">
   <xs:element name="Branch"  sql:SQLName="branch">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Department" type="DepType"  sql:SQLName="office"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="DepType"  sql:SQLType="officeType">
   </xs:complexType>
</xs:schema>
```

Note the Oracle namespace (with the sql prefix) and annotations for tables (sql:SQLName) and ADTs (sql:SQLType)

Generating a DAD File for IBM DB2

IBM DB2 v8.1 (or higher) is a database server with an add-in for XML storage and retrieval called IBM DB2 Extender. XML data (elements, attributes) are mapped to relational data (tables, columns) through Document Access Definition files (.DAD).

There are three types of DAD files:

Storage Type	Description
Xcolumn	Column mapping - the Root element is mapped to a table, and its attributes or child elements are mapped to columns identified by an XPath
Xcollection	SQL mapping - the DAD file starts with a SQL statement for the table mapped to the Root element, and each child element or attribute is mapped to a column or a table name
Xcollection	RDB mapping - a Relational Database node, with a table and a column name, is associated with each attribute or child element of the Root element

An XML model targeted with DTD allows you to generate DAD files for IBM DB2.

1. Map an XSM to a PDM. You can do this manually or by generating an XSM from a PDM (or a PDM from an XSM) but we recommend that you use the XML Builder Wizard (see *Mapping database objects to an XML schema via the XML Builder Wizard* on page 97)
2. [if you do not use the wizard] Attach the IBM DB2 DAD extension file. To enable these extensions in your model, select **Model > Extensions**, click the **Import** tool, select the `IBM DB2 DAD` file (on the **XML in Database** tab), and click **OK** to attach it..
3. Further specify the mappings with extended attributes (see *DB2 Extended Attributes for Global Elements* on page 108).
4. [optional] Click the **Preview** tab of the Root element property sheet, and select the *DB2XMLExtender.DAD File* tab to preview the DAD file. If the DAD File tab is not available, click the **Select Generation Targets** tool to select IBM DB2 DAD in the **Targets** list and click **OK**.
5. Generate the annotated schema (see *Generating a DB2 DAD file* on page 108).

DB2 Extended Attributes for Global Elements

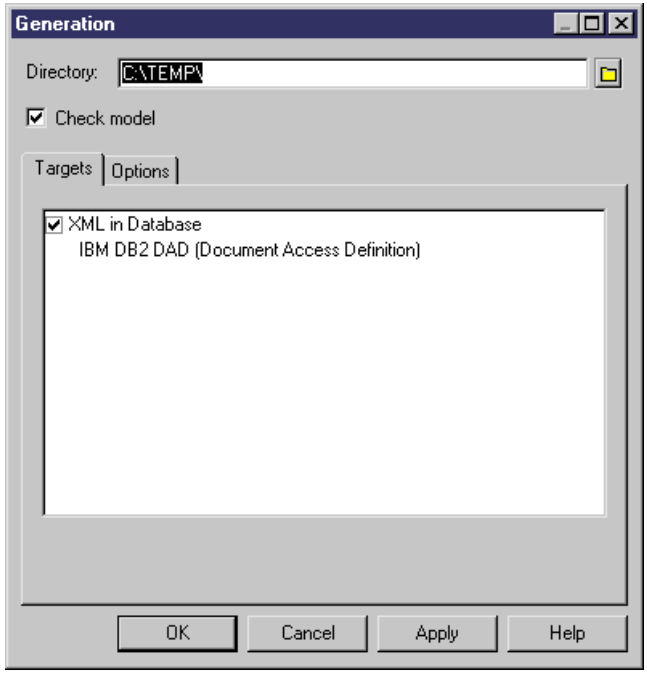
You can set extended attributes on global elements to reinforce their mapping to tables and columns, by opening their property sheets and clicking the Extended Attributes tab.

Extended attribute	Description
Database	Name of the database
DTDID	ID added to the DTD_ref system table in DB2 XML Extender
Login	Name of the logged-in user
MappingType	Type of mapping for a collection
NamespaceNode	Text zone where each line describes a namespace couple (name = value). The separator character is '='
Password	Password of the logged-in user
PathGeneration	Generation path
ProcessInstruction	A text zone that enables the user to enter some instruction
SideTableID	Identifier of the side table (optional)
SideTableName	Name of the side table
StorageName	If StorageType is Xcolumn, then it is the name of the sidetable column
StorageType	Type of storage (Xcollection or Xcolumn)

Generating a DB2 DAD File

You generate the DAD file by selecting it as an additional target for standard schema generation.

1. Select **Language > Generate *schema*File** to open the Generation dialog box.
2. Specify the directory in which to generate the file and select the XML in Database target on the Targets tab.



3. [optional] Click the Options tab, and set any appropriate generation options:

Option	Description
Character ending an instruction	Character ending instructions in the SQL file for stored procedures
Generates procedures deployment	Generation of a SQL script for stored procedures enabling XML data storage and facilitating XML data retrieval
Path of DAD.dtd	Path of the DTD file installed with IBM DB2 Extender and describing the specific syntax of DAD files
Schema validation	Validation tag in the DAD files to check the conformity of DAD files with the DAD syntax

4. Click OK to begin the generation.

   The Result dialog box is displayed with the path of the generated DAD, DTD and SQL files.

5. Click Edit to open the generated DAD file in your associated editor:

   - Extract of a DAD file defined with *Xcollection* as StorageType, and *RDB* as MappingType:

```
<!DOCTYPE DAD SYSTEM "E:\dad.dtd">
<DAD>
<validation>YES</validation>
<Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Root SYSTEM "C:\TEMP\DTD_files\CorporateMembership.dtd"</doctype>
<root_node>
<element_node name="Root">
   <element_node name="DEPARTMENT">
      <attribute_node name="DEPNUM">
         <RDB_node>
            <table name="DEPARTMENT"/>
            <column name="DEPNUM" type="INTEGER"/>
         </RDB_node>
      </attribute_node>
      <attribute_node name="DEPNAME">
         <RDB_node>
            <table name="DEPARTMENT"/>
            <column name="DEPNAME" type="VARCHAR(254)"/>
         </RDB_node>
      </attribute_node>
      <element_node name="EMPLOYEE">
         <attribute_node name="DEPNUM">
            <RDB_node>
               <table name="EMPLOYEE"/>
               <column name="DEPNUM" type="INTEGER"/>
            </RDB_node>
         </attribute_node>
```

- DAD file defined with *Xcolumn* as StorageType:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "E:\dad.dtd">
<DAD>
<validation>YES</validation>
<Xcolumn>
<table name="DEPARTMENT" key="DEPNUM" orderBy="DEPNUM, , DEPNAME">
   <column name="DEPNUM"
           type="INTEGER"
           path="/DEPARTMENT/@DEPNUM"
           multi_occurrence="NO"/>
   <column name="DEPNAME"
           type="VARCHAR(254)"
           path="/DEPARTMENT/@DEPNAME"
           multi_occurrence="NO"/>
</table>
<table name="EMPLOYEE" key="EMPID" orderBy="EMPID, DEPNUM, FIRSTNAME, LASTNAME">
   <column name="DEPNUM"
           type="INTEGER"
           path="/DEPARTMENT/EMPLOYEE/@DEPNUM"
           multi_occurrence="NO"/>
   <column name="EMPID"
           type="INTEGER"
           path="/DEPARTMENT/EMPLOYEE/@EMPID"
           multi_occurrence="NO"/>
   <column name="FIRSTNAME"
           type="CHAR(1)"
           path="/DEPARTMENT/EMPLOYEE/@FIRSTNAME"
           multi_occurrence="NO"/>
   <column name="LASTNAME"
           type="CHAR(1)"
           path="/DEPARTMENT/EMPLOYEE/@LASTNAME"
           multi_occurrence="NO"/>
</table>
</Xcolumn>
</DAD>
```

Index

A

All (group particle) 29
annotated schema
    Microsoft SQL Server 2000 100
    Oracle 9i2 103
annotation 67
    application information 67
    check model 95
    create 68
    documentation 67
    global 67
    local 67
    properties 68
any 33
    create 34
    namespace 33
    process contents 33
    properties 34
any attribute 40
    namespace 40
    process contents 40
application information 67
attribute 35
    check model 89
    create 37
    properties 37
attribute group 52
    check model 90
    create 53, 54
    properties 52, 54
    reference 52
    stereotype 52
AttributeType (XDR) 37

B

base type 61
business rule (XSM)
    define 74

C

check model 83
    annotation 95
    attribute 89
    attribute group 90
    complex type 86
    data source 85
    element 87
    entity 85
    extension 93
    group 88
    group particle 83
    import 90
    include 86
    key 91
    keyRef 92
    model 84
    namespaces 84
    notation 89
    redefine 91
    restriction 94
    shortcut 84
    simple type 86
    simple type list 94
    simple type union 95
    target namespace 84
    unique 93
child element 29
Choice (group particle) 29
code
    preview 8
complex content 60
complex type 56
    check model 86
    complex content 60
    create 57
    global 56
    local 56
    properties 57
    simple content 60
constraint 41
    create 43
    properties 44
context node 45

D

DAD file 107

data source 85
data type
    attribute check 89
    complex type 56
    element check 87
    extension 61
    external shortcut 17
    simple type 55
    simple type list 66
    simple type list check 94
    simple type union 67
    simple type union check 95
database
    DAD file 107
    IBM DB2 107
    Microsoft SQL Server 2000 100
    Oracle 9i2 103
    SQL/XML queries 97, 99
    XML in database 97
derivation 61
    extension 61
    simple type list 66
    simple type union 67
diagram 13
display preferences 9
documentation
    annotation 67
DTD 1

E

element 18
    check model 87
    child element 29
    create 21
    general properties 21
    group 48
    parent element 29
embedded type 66
entity 70
    check model 85
    create 71
    properties 71
extended dependency 12
extension 11
    check 93
    derivation 61
extension file 11

F

facet 62
field 41, 45
    stereotype 45
    XPath 45

G

generate
    DTD file from XML model 75
    XDR file from XML model 75
    XSD file from XML model 75
generate pdm 79
generate xsm 79
global objects 27
group 48
    check model 88
    create 50
    properties 48, 50
    reference 48
    stereotype 48
group particle
    All 29
    check model 83
    Choice 29
    create 30
    properties 31
    Sequence 29

I

IBM DB2 107
identity constraint 41
    field 45
    key 41
    keyRef 41
    selector 45
    unique 41
import 72
    check model 90
    create 73
    properties 73
include 72
    check model 86
    create 73
    properties 73

K

key 41

field 41
properties 41
selector 41
stereotype 41
key (check model) 91
keyRef 41
check model 92
field 41
properties 41
selector 41
stereotype 41

L

link 16
child object to complex type 59
child object to element 26
child object to group 51
child object to group particle 32
child object to parent object 16
local objects 27

M

manipulate XML objects graphically 27
member types 67
mixed (content) 60
model
check model 84
copy XSM 4
create 4
model options 9
preview code 8
properties 5
share XSM 4
XML 1
XML language 4
model options 9
modeling environment
customize 9

N

namespace 33, 40
node
context node 45
root node 45
notation 69
check model 89
create 70
properties 70

O

Oracle 9i2 103

P

parent element 29
preview code 8
process contents 33, 40

R

RDB 97
redefine 72
check model 91
create 73
properties 73
reference 41, 48, 52
restriction
check model 94
detail properties 62
reverse engineering
options 77
target models 77
to existing XML model 78
to new XML model 77
XSD, DTD or XDR file to existing XML model 77
XSD, DTD or XDR file to new XML model 77
root node 45

S

schema 1
selector 41, 45
stereotype 45
XPath 45
Sequence (group particle) 29
shortcut
check model 84
managing external shortcuts through references and data types 17
simple content 60

simple type 55
  check model 86
  create 55
  derive by list 66
  derive by union 67
  derived by list 55
  derived by restriction 55
  derived by union 55
  list check 94
  properties 56
  union check 95
SQL/XML query (in XML model) 97, 99
stereotype 41, 45
  attribute group 52
  constraint field 45
  group 48
  unique constrainté 41

U

union
  member types 67
  properties 67
unique 41
  check model 93
  field 41
  properties 41
  selector 41
  stereotype 41

X

XDR 1
  any 33
  AttributeType 37
xem 11
XML
  diagram 13
  model 1
  objects 14
XML diagram
  attribute 35
  create 17
  element 18
  entity 70
  import 72
  include 72
  notation 69
  redefine 72
XPath 45
  abbreviated syntax 45
  expressions 45
XSD 1
XSM 1
  changing 10
  check model 83
  create 4
  edit definition file 10
  functional overview 1